PUBLIC
SAP Cloud Applications Studio
Document Version: 2002 – 2020-08-28

SAP Cloud Applications Studio

© 2020 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

Content

1 Working with the Studio Library. 8

2 Upgrade Information for Customer-Specific Solutions. 17

Customer's Tenant. 17

Development Tenant. 18

3 Log on to the Repository. 19

4 What's New in the SAP Cloud Applications Studio. 21

5 Getting Started. 23

5.1 Getting Started with the Studio. 23

Update Your Language Preference. 25

5.2 About the SAP Cloud Applications Studio. 25

Features. 25

5.3 Feature Matrix. 27

5.4 First Steps. 30

Installing and Updating the SAP Cloud Applications Studio. 30

User Roles. 32

User Setup Quick Guide. 33

5.5 Introduction to the Integrated Development Environment. 42

Working in the Integrated Development Environment. 42

My Solutions. 48

Repository Explorer. 49

Execute a Query to Explore a Business Object. 52

Solution Explorer. 54

Add New Item Dialog Box. 58

Properties Window. 67

Trace Explorer. 69

Working with the Wizards. 70

Icon Library (Studio). 72

Icon Library (UI Designer). 75

5.6 Public Solution Model (PSM). 77

About the Public Solution Model (PSM). 77

5.7 Introduction to Business Object Modelling. 81

Business Object Modeling. 81

Business Object Modeling: Data Types. 92

Look Up the Representation Term of a Data Type. 96

Business Object Categories. . . . 97
5.8 Example: Build a Simple Solution. . . . 99
Example: Building a Solution. . . . 99
Sample Code. . . . 114
Further Development. . . . 116
5.9 Using the Localization Toolkit (SAP Business ByDesign). . . . 139

6 Administration and Lifecycle Management. . . . 141
6.1 Overview of Administration and Lifecycle Management. . . . 141
6.2 Lifecycle Management. . . . 143
6.3 Quality Criteria for Customer-Specific Solutions. . . . 144
Performance. . . . 144
Security. . . . 145
Supportability. . . . 147
6.4 Developing on a Customer's Tenant. . . . 149
Lifecycle Management for Customer-Specific Solutions on a Customer's Test Tenant. . . . 149
Create a Patch on a Customer's Test Tenant. . . . 152
What You Need to Consider When a Change Project Is on the Same Test Tenant. . . . 154
6.5 Developing on Your Development Tenant. . . . 155
Lifecycle Management of Customer-Specific Solutions on Your Development Tenant. . . . 155
Create a Patch on Your Development Tenant. . . . 157
Switch a Customer Assignment. . . . 158
6.6 Administration. . . . 159
Administration Quick Guide. . . . 159
Implementation Manager Quick Guide. . . . 162
Migration Quick Guide. . . . 168
6.7 Solutions, Solution Templates, and Patches. . . . 171
Solution Templates Quick Guide. . . . 171
Solution Templates. . . . 174
Create a Solution Template. . . . 175
Create a Solution. . . . 177
Patches. . . . 178
Enable or Disable Your Solution in the SAP Cloud Solution. . . . 182
6.8 Bug Fix in Production and Pre-production Tenants. . . . 184
6.9 Maintenance Mode. . . . 185

7 Scripting Languages. . . . 188
7.1 Overview of Scripting Languages. . . . 188
7.2 Scripting Language for the Studio. . . . 189
Scripting Language Reference. . . . 189
Business Object Definitions. . . . 191
Business Object Extension Definitions. . . . 218

Business Logic. . . . 235
SAP Reuse Libraries. . . . 274
SAP Built-In Functions. . . . 320
Basic Data Types. . . . 329
7.3 Front-End Script for the UI Designer. . . . 335
Front-End Script Reference. . . . 335
Lexical Structure of Front-End Scripts. . . . 338
Syntax of Front-End Scripts. . . . 339
Front-End Code Snippets. . . . 343
Interacting with Runtime Data. . . . 349
Global Variable. . . . 354

8 Developer Desktop. . . . 360
8.1 Overview of the Developer Desktop. . . . 360
8.2 Business Configuration. . . . 362
Business Configuration Quick Guide. . . . 362
Business and Technical Background. . . . 370
Tasks. . . . 375
8.3 Business Objects. . . . 398
Business Objects Quick Guide. . . . 398
Business and Technical Background. . . . 403
Tasks. . . . 418
8.4 Extensibility. . . . 438
Business Object Extensions. . . . 438
Process Extension Scenarios. . . . 459
Create an ODATA Service Extension. . . . 463
Enhancement Implementations. . . . 464
8.5 Reuse Libraries. . . . 468
Reuse Library Quick Guide. . . . 468
Create Your Reuse Function. . . . 470
8.6 Code List Data Type. . . . 472
Create a Code List Data Type. . . . 472
Example: Create a Code List Data Type (Purchase Order Request). . . . 473
8.7 Data Protection and Privacy. . . . 476
Enable Data Disclosure. . . . 476
Enable Data Removal. . . . 480
Enable Read Access Logging. . . . 483
Enable Personal Data for Extension Fields. . . . 485
8.8 Approval Process. . . . 486
Approval Process Quick Guide. . . . 486
Example: Create an Approval Process (Price Request). . . . 487
Business and Technical Background. . . . 491

Tasks. . . . 495
8.9 Notification Rules. . . . 501
Enable Notification Rules. . . . 501
Example: Enable Notification Rules (Price Request). . . . 504
8.10 Print Forms. . . . 506
Print Forms Quick Guide. . . . 506
Associating an .xdp File with Adobe LiveCycle Designer. . . . 508
Create a Print Form. . . . 508
Test a Print Form. . . . 510
8.11 Analytics. . . . 511
Analytics Quick Guide. . . . 511
Business and Technical Background. . . . 515
Tasks. . . . 527
8.12 Mass Data Run. . . . 537
Mass Data Run Quick Guide. . . . 537
Business and Technical Background. . . . 538
Mass Data Run. . . . 538
Tasks. . . . 542
Create a Mass Data Run. . . . 542
Assign MDR Work Center View to UI. . . . 544
Test the Mass Data Run. . . . 544
Example. . . . 546
Example: Changing the Status of a Price Request (Mass Data Run). . . . 546
8.13 Mashups. . . . 551
Mashups Quick Guide. . . . 551
Business and Technical Background. . . . 553
Tasks. . . . 554
8.14 Service Integration. . . . 575
Service Integration Quick Guide. . . . 575
Condition Evaluation for Internal Communication. . . . 577
Tasks. . . . 579
8.15 Web Services. . . . 586
Web Services Quick Guide. . . . 586
Web Services. . . . 591
Web Service Operations. . . . 596
External Web Service Integration. . . . 604
Tasks. . . . 606
8.16 Report an Incident. . . . 628
8.17 Debugging. . . . 629
Debugging Quick Guide. . . . 629
Business and Technical Background. . . . 631

8.18 Tracing. . . . 639
Tracing Quick Guide. . . . 639
Tracing. . . . 641
Trace Class and Methods. . . . 643
Tasks. . . . 646
8.19 Dump Analysis. . . . 649
Dump Analysis Quick Guide. . . . 649
8.20 Translation. . . . 651
Translation Quick Guide. . . . 651
Business and Technical Background. . . . 654
Tasks. . . . 658

9 User Interface Designer. . . . 662
9.1 User Interface Designer Overview. . . . 662
9.2 Getting Started with the User Interface Designer. . . . 664
Features of the User Interface Designer. . . . 664
Working with the Checkmate Tool. . . . 668
9.3 Modeling Floorplans. . . . 694
Floorplan Modeling Quick Guide. . . . 694
Tasks. . . . 697
9.4 Modify Your Floorplans. . . . 719
Floorplan Modification Quick Guide. . . . 719
Business and Technical Background. . . . 724
Tasks. . . . 731
9.5 Modify an SAP Floorplan. . . . 755
Change Transactions Quick Guide. . . . 755
Business and Technical Background. . . . 758
Tasks. . . . 761
9.6 User Interface Modifications for Mobile Applications. . . . 774
Mobile Solutions. . . . 774
Tasks. . . . 778

10 Style Guides and Reference Guides. . . . 792
10.1 Overview of Style Guides and Reference Guides. . . . 792
10.2 Style Guides. . . . 792
Style Guide for Writing System Message Texts. . . . 792
10.3 Reference Guides. . . . 821
Reference Guide for Documentation for Cloud Solutions from SAP. . . . 821

11 Copyright / Legal Notice. . . . 840

12 Glossary. . . . 841

13 Important Disclaimers on Legal Aspects. 862

1 Working with the Studio Library

Overview

In this library you can view reference documentation about the SAP Cloud Applications Studio (“studio”). The library contains reference documentation to help you get started with the studio, as well as information related to quality assurance, information for studio administrators, and detailed information on development activities in the studio. The library also contains reference documentation for the user interface designer (UI designer), and references the general concepts and tools in the SAP cloud solutions.

In the studio, select *Help* > *SAP Cloud Applications Studio Help*. The library opens in your browser and is displayed using a content player. This option does not offer a function to search the library.

> **i Note**
>
> This documentation does not describe the standard features and technologies in Microsoft Visual Studio. For information about Microsoft Visual Studio, see http://www.microsoft.com .

> **⚠ Caution**
>
> Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

Getting Started

First Steps

Learn about the following:

- Installing SAP Cloud Applications Studio [page 30]
  You have to install the latest version of the SAP Cloud Applications Studio ("studio") from the SAP Service Marketplace. This document takes you through the requirements and steps to install the studio

  > **i Note**
  >
  > Prior to working in the studio, you must change the initial password provided to you in the SAP cloud solution.

- User Setup Quick Guide [page 33]
  Before you can begin working in the studio, the initial user has to create the required users for the studio and for the SAP cloud solution. This quick guide explains the entire process from creating service agents, requesting users for these service agents, assigning an ID and an initial password to each user, and assigning the required work centers. In addition, it also explains the process to create business users for testing purposes in the SAP cloud solution.
- User Roles [page 32]
  Find out about the different user roles in the studio:
  - For development purposes: Studio Administrator and Developer
  - For testing purposes: Business User
- **Features of the Studio**
  The studio enables you to adapt and enhance the solution capabilities of the SAP cloud solution. Using the studio, you can develop, deploy, and test specific add-on functionality.
  For an overview of the studio features, see About SAP Cloud Applications Studio [page 25].

  > **⚠ Caution**
  >
  > This documentation covers the studio as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every SAP cloud solution. Before using a particular feature, you should consult the Feature Matrix to determine whether the feature is available for the SAP cloud solution for which you are creating a solution. If the feature is available, it is marked with an X in the relevant column of the matrix.
  >
  > To check which features are available for which SAP cloud solution, see Feature Matrix [page 27].

- Localization Toolkit (SAP Business ByDesign only) [page 139]
  The localization toolkit is a set of development tools and instructions that enable SAP partners to extend and adapt the capabilities of the SAP Business ByDesign solution to provide a locally-compliant solution for customers.

Get an Introduction to Business Object Modeling and the Public Solution Model

- Introduction to Business Object Modeling [page 81]
  the SAP cloud solutions are built using the concept of business objects to model the business environment.
- Public Solution Model [page 77]
  The SAP public solution model (PSM) contains all entities in the SAP cloud solution that are released for use by external consumers. External consumers can be partners who develop solutions on top of the SAP cloud solution, such as add-ons and integration scenarios, or administrators in the SAP solution, who, for example, use SAP data sources to create new reports.
- Repository Explorer [page 49]
  The *Repository Explorer* tool window displays SAP content that is released with the public solution model (PSM) such as business objects, data types, enhancement options, inbound service interfaces, and reuse UIs. You can explore the content to gain an understanding of the repository structure and the business context and use of SAP content, which you can reuse and enhance in your solution.

Solution Examples

This documentation contains various solution examples. They include coding and show you how to use particular functions of the studio. You also learn how to create a complete solution in the studio, configure the screens in the UI designer and how to use the functions in SAP's cloud solution.

Build a solution to manage the allocation of car parking spaces to employees:

- Example: Building a Solution [page 99]

Try out particular functions of the studio:

-
- Example: Create a Code List Data Type (Purchase Order) [page 473]
- Example: Enable Notification Rules (Price Request) [page 504]
- Example: Changing the Status of a Price Request (Mass Data Run) [page 546]
- Example: Calculate Duration of Campaign (References to Customer-Specific Fields) [page 455]
- Example: Create an Approval Process (Price Request) [page 487]
- Example: Consume a REST Service [page 616]

Administration and Lifecycle Management

Get Informed About Administration and Lifecycle Management of Customer-Specific Solutions

- Administration Quick Guide [page 159]
  The *Administration* toolbar provides access to a number of administrative tasks, such as creating a new solution, deleting a solution, and managing user sessions.
- Implementation Manager Quick Guide [page 162]
  You use the *Implementation Manager* to manage the lifecycle of customer-specific solutions.
- Lifecycle Management [page 143]
  The studio supports the full end-to-end lifecycle management of customer-specific solutions. This includes developing, testing, and deploying your solution, as well as creating patches to deliver updates to your customer.
- Developing on a Customer's Tenant [page 149]
  You can develop a customer-specific solution on the customer's test tenant. The finalized solution is then uploaded and activated on the customer's production tenant.
- Developing on Your Development Tenant
  - Lifecycle Management on Your Development Tenant [page 155]
    You can develop a customer-specific solution on your development tenant and after having tested, assembled, and downloaded the solution, you can upload it to the customer's tenant.
  - Switch a Customer Assignment [page 158]
    If you are developing customer-specific solutions on your development tenant, you can create solutions for more than one customer. For example, if you are working on a solution for customer A, you can save your solution and then switch to work on the solution for customer B. For this, you need to switch the customer assignment. Afterwards, in the My Solutions tool window in the studio, only the solutions of customer B are listed and you can create new solutions only for this specific customer.
- Quality Criteria [page 144]
  Customer-specific solutions for the SAP cloud solution are installed in a multi-tenant environment. The solution provider needs to ensure that all solutions are of the highest quality and have the lowest possible Total Cost of Ownership (TCO). SAP requests you to conform to the criteria described in this document in order to ensure the quality of your solution.

Get Informed About Solution Templates, Patches, Maintenance Mode, and Upgrade Information

- Solution Templates [page 171]
  With a solution template you can organize common development content that you want to reuse for customer-specific solutions. Solution templates enable you to rapidly start the development of customer-specific solutions, for example, for a specific industry
- Patches [page 178]
  A patch is needed if you want to make changes or corrections to a customer-specific solution after it has been assembled. The customer-specific solution is then in maintenance mode and you can only make restricted changes for certain content types.
- Maintenance Mode [page 185]
  A customer-specific solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is Assembled. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.
- Upgrade Information [page 17]
  Upgrade information is relevant if your customer's system is upgraded to a new release. The upgrade is maintained by SAP to ensure high quality upgrade of the customer's production tenant. The upgrade does not have any impact on your customer-specific solution. However, be aware of the downtime period during the upgrade and follow the rules listed in this document.

Scripting Languages

Get Informed About Scripting Languages for the Studio

- Scripting Language Reference [page 189]
  The scripting language is used to define business objects and business object extensions as well as to implement the business logic for cloud partner solutions. This lightweight language is easy to learn and to use with its focused set of features.
  The keywords, methods and functions of the scripting language are described in separate sections according to the task you want to perform:
  - Syntax for Business Object Definitions [page 191]
    You use these keywords and functions to define the structure and interface of business objects.
  - Syntax for Business Object Extension Definitions [page 218]
    You use these keywords and functions to define the structure and interface of business object extensions.
- Syntax for the Implementation of Business Logic [page 235]
  You use these methods and keywords to implement the business logic of your solution. For this task, you can also use predefined built-in functions and reuse services:
  - Built-In Functions [page 320]
  - Basic Reuse Libraries [page 274]
  - Reuse Libraries for Business Areas [page 308]
- Basic Data Types [page 329]
  The scripting language supports basic data types such as string, numeric and Boolean values. In addition, the existing metadata of the SAP underlying cloud platform for core data types (CDTs) and global data types (GDTs) is made available.

Get Informed About Scripting Languages for the UI Designer

- Front-End Script for the User Interface Designer [page 335]
  The front-end script is the programming language used in the UI designer to define a set of rules and the logic of the behavior of UI controls. For example, you can write a front-end script to enable an edit button only when a field is selected on the user interface.

Developer Desktop

Choose a Help Topic for the Developer Desktop

- Business Configuration [page 362]
  Learn how to create and change business topics, business options, and business configuration sets for your solution. Also find out how to configure a business configuration view and assign it to the *Business Configuration* work center and how to check your content in the business adaptation catalog.
  In addition, find information on the tools available in the SAP cloud solution to assist you in creating and editing your business configuration content.
- Business Objects [page 398]
  Learn how to create a business object, define its nodes and relevant elements, create script files, activate your business object, and generate the screens for your business object.
- Extensibility
  - Business Object Extensions [page 438]
    Learn how to create a business object extension, define business logic for a business object extension, screens, reports, and the Enterprise Search.
  - Process Extension Scenarios [page 459]
    A process extension scenario links the data from one business context to other related business contexts. Learn how to create a process extension scenario from a list of predefined extension scenarios.
  - Enhancement Implementations [page 464]
    SAP provides enhancement options based on SAP technology that allow you to enhance the standard behavior of specific applications of the SAP cloud solution without modifying the standard solution. Learn how to create enhancement implementations of SAP enhancement options.
- Reuse Libraries [page 468]
  Learn how to create and define your own reuse libraries and reuse functions to organize complex code in your solution or to reuse your code in script files.
- Code List Data Type [page 472]
  Learn how to define a code list data type for your solution and use it when you create your custom business object or business object extension.
- Notification Rules
- Analytics [page 511]
  Learn how to create and change data sources, and create and manage reports, and get informed about analytical report content, transformation rules, and aggregation.
  In addition, find information on the tools available in the SAP cloud solution to assist you in defining your reports.
- Mashups [page 551]
  Learn about creating mashups using the studio and the UI designer.
- Service Integration [page 575]
  Learn how to configure internal communication to communicate with business objects in different deployment units, and create XML file input to import external XML data into a business object.
  In addition, find information on the tool for file input in the SAP cloud solution.
- Web Services [page 586]
  Learn how to create, modify, authorize and test a Web service, and how to remove authorization to a Web service.
- Mass Data Run [page 538]
  Learn how to create a mass data run for your custom business object.
- Print Forms [page 508]
  Learn how to create a print form, add a preview button to it, and test the print form. In addition, find information on the tools available in the SAP cloud solution to assist you in creating and editing your print forms.
- Approval Process [page 493]
  Learn how to create a default approval process, assign it to the UI, and test it.
  In addition, find information on the tools available in the SAP cloud solution to assist you in creating and editing your approval process.

Choose a Help Topic for the Developer Desktop

Learn how to enable your custom business object for notification rules.

Find out about Debugging, Tracing, Dump Analysis, and Translation

- Debugging [page 629]
  Learn how to use the debugger to observe the runtime behavior of your solution and to locate logic errors. The debugger works with your script files and allows you to break the execution of your solution at specified breakpoints to examine your code and to display values of variables defined in your solution.
- Tracing [page 639]
  Learn how to use tracing to record information about your code as it is being executed. From the *Trace Explorer*, you can run a local trace, run a trace for a business user in the SAP cloud solution, use incident traces for troubleshooting purposes, and view the results of a recorded trace.
- Dump Analysis [page 649]
  The *Dump Analysis* tool window presents dump information in a reader friendly format. Learn how the tool window supports you in finding the root cause of a dump.
- Report an Incident [page 628]
  If you encounter a problem during development, you can report an incident in the SAP cloud solution to request help in solving the problem.
- Translation [page 651]
  Find out how you can translate your solution content into the languages supported by the SAP cloud solution.

User Interface Designer

Get Informed About the UI Designer

- Introduction to the UI Designer [page 662]
  The user interface designer (UI designer) is integrated into the studio, and you can use it to create new screens, or modify the screens provided by SAP. The UI designer can also be used to design screens for use on mobile devices.
- Features of the UI Designer [page 664]
  Get a quick overview of the common features available to develop a user interface.

Check Out How to Model and Modify Floorplans

- Floorplan Modeling [page 694]
  Get an overview of the steps involved in modeling different types of floorplans.
- Floorplan Modification [page 719]
  Get informed about the modifications and enhancements you can make to the floorplans you created.
- Change Transactions [page 758]
  Get an overview of the changes you can make to SAP floorplans. These changes are termed as change transactions and need anchors to be executed.
-
  Get an overview about how to adapt floorplans for use on mobile devices and the steps involved in configuring navigation between floorplans.

Style Guides and Reference Guides

Check Out the Style Guides	Check Out the Reference Guides
Style Guide for Writing System Message Texts [page 792] This style guide provides you with the standards and guidelines for writing system messages to be displayed on the user interface (UI) of your solution. It also provides the standards and guidelines for writing system messages that are relevant for SAP Support; these messages are not displayed on the UI. In addition, this style guide provides guidance on writing long-text documents providing additional information to a system message that helps the user to solve very complex error situations. These long-text documents can be incorporated into the Solution Documentation.	Reference Guide for Documentation of Cloud Solutions from SAP [page 821] This reference guide provides an overview of the main reference documentation types specific to the SAP cloud solution.

Discussion Forum

Ask your peers questions and search for answers.

You can find the discussion forum in the SAP Community at SAP Cloud Applications Studio Community.

2 Upgrade Information for Customer-Specific Solutions

The following information is relevant, if the tenants of your customer are upgraded to a new version. The steps you need to take to upgrade your customer-specific solution differ according to the tenant type on which you are developing.

Customer's Tenant

SAP maintains the upgrade to a new version to ensure high quality upgrade of the customer's production tenant and test tenants.

What you need to consider before the upgrade

The upgrade does not have any impact on your customer-specific solution. However, during the upgrade process you **cannot** continue development.

> **i** Note
>
> Please take into account that you cannot access the customer's tenant during the upgrade preparation phase (this is one week before the downtime period of the upgrade) and during the upgrade period itself.

What you need to consider after the upgrade

After the tenants have been upgraded, you continue working on your solution in the studio version that matches the SAP upgraded cloud solution version. For example, if the SAP cloud solution has been upgraded from version 1305 (May 2013) to version 1308 (August 2013), you continue working with the studio version 1308.

Please be aware of the following:

- If the solution still has the status *In Development* and you have activated its items before the upgrade, you must open the solution after the upgrade in the corresponding studio version and reactivate the solution items. Only then can you download the solution to upload it to the upgraded tenant.
- If you have downloaded a solution just before the upgrade, and then want to upload this solution to the upgraded tenant, you must first open the solution in the studio version matching the SAP upgraded cloud solution version. Then you must download it again, so that your solution is compatible with the version of the SAP cloud solution. Now you can upload your solution to the upgraded tenant.

Development Tenant

SAP does **not** upgrade your development tenant. However, if your solution has been deployed in a customer's tenant, it will automatically be upgraded when SAP upgrades that tenant. You can then download the upgraded version of your solution, upload it to the next studio version, and then create a patch for this solution.

You can upload solutions that have not yet been deployed in a customer's tenant as well as solution templates to the next studio version or the subsequent studio version, for example:

If you have created your solution or solution template in this version ...	... you can upload it to this version	... and to this version
SAP Solutions OnDemand Studio 1302 (February 2013)	SAP Solutions OnDemand Studio 1305 (May 2013)	SAP Cloud Applications Studio 1308 (August 2013)

3 Log on to the Repository

Overview

Before you start working in the studio, you must connect to the repository, which contains any solutions that you develop as well as the SAP public solution model (PSM) content.

Prerequisites

You have a user and password for the repository. For more information, see User Setup Quick Guide [page 33].

Procedure

1. Open the studio.
2. The *Connect to Repository* dialog box is displayed. Select the *System* that contains the repository to which you want to connect.

   > **i Note**
   >
   > In most cases, only one *System* is available; this is your development repository.
   >
   > To edit the repository system connection, from the *Administration* menu, click the pencil icon (✎) to the right of the *System* field or select *Options and Settings* and edit the settings on the *Connectivity* tab.

3. Enter your *User ID* and *Password*; then click *OK*.
   There is an information icon to the right of each entry field. When you click one of them, the system displays information about the corresponding entry field.

   To change your password, click the pencil icon (✎) to the right of the *Password* field. The SAP cloud solution opens where you can change your password: Enter your credentials and click *Change Password*. You are then prompted to enter a new password.

   > **i Note**
   >
   > If you face an error, it may be due to any of the following reasons:
   >
   > - While logging to the SAP Cloud Application Studio for the first time , you have not maintained the correct proxy settings. To solve the issue, follow the steps below:
   >   1. Go to SAP Cloud Application Studio *Connect Repository* and click on *Edit System Entries*.
   >   2. Go to the *Connectivity* tab and select your host system.
   >   3. Choose your *Proxy Settings* as **http://proxy:8080** and select *Use HTTP Proxy*.
   >   4. Save the settings and then click on *OK*.
   > - Network Issues

This is generally a temporary issue. As a work around you can switch the network connection and change it back to the previous one.

- Incorrect URL
  The host URL should be the tenant URL only.

Result

After you have logged on to the repository, any solutions that you have created appear in the *My Solutions* tool window. This is what you can do from here:

- You can open a solution to view and work on the solution content in the *Solution Explorer*.
- You can create a new solution. For information about creating solutions, see Create a Customer-Specific Solution [page 177].
- When you hover over a solution in the *My Solutions* window, a quick info text is displayed. It shows the name, description, type, and status of the solution and tells you, if the solution is a patch solution. This is an example of such a quick info text:

Name:	`Y77VXGW3Y_`
Description:	`My Test Solution`
Type:	`Customer-Specific Solution`
Status:	`In Development`
Patch Solution:	`No`

Related Information

My Solutions [page 48]
Solution Explorer [page 54]
Repository Explorer [page 49]

4 What's New in the SAP Cloud Applications Studio

This document provides information about what is new or changed in the **SAP Cloud Applications Studio** version 2002 (February 2020).

> ⚠ Caution
>
> Any software coding or code lines/strings ("Code") provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

What is New in All Cloud Solutions

In version 2002 of the SAP Cloud Applications Studio, the following features are new:

New Content for Deletion Patch

You can now delete the following content from a productive solution under maintenance:

- Communication scenario definition (CSD)
- Web service definition (WSID)
- Enhancement Implementation (BAdI)

This is useful if you want to use the standard solution provided by SAP and delete the enhancement implementation customized for you.

To do this, create a deletion patch and delete the communication arrangement from *Administrator* > *General Settings* > *Integration*.

> ℹ Note
>
> - It is mandatory to deploy the deletion patch in production tenant. For consecutive deletion patches, you must deploy the latest one.
> - It is not possible to transport deletion patch using Transport Management.

Display, Compare, and Restore Previous Version of ABSL Codes

You can now display, compare, and restore previous versions of ABSL code within the current patch version.

This is possible even if multiple developers have worked on it.

The ABSL code version is compared only within the current patch and not across the patch versions.

> **i** Note
>
> Restore is possible only for the in-development solution.
>
> It's not possible to restore code in production bug fix scenario.

Additional Enhancements

- BC View URL: You can generate a BC View URL even if the BC View is created in a different patch than the patch in which the BCO and BC set is created.
  Customers create a solution which has a BCO (Business Configuration Object) and a BC Set created on this BCO in one of the version and the solution is deployed to production. In a subsequent patch in the source, add a BC view for the mentioned BCO. However during deployment of this subsequent patch with the BC view, the BC View URL is not generated.
- Reuse Library Functions of Quantity Conversion: The Add and Subtract functions of the Quantity Conversion library now supports a specific UoM (Unit of Measure). When specified, these functions return the result in the target UoM. For example, you can add two quantity values, one in hour, and the other in minutes, and get the result of addition in seconds.
- Custom Object Builder: Earlier, when a web service was generated for a custom object, and if the name of a query selection parameter had a length of more than 120 characters, the web service generation failed without displaying an error message. This is fixed, and now either selection parameter in the web service gets an abbreviated name, or in some cases if the aliasing is not possible then an error message is displayed advising the administrator to fix the name of the element in the custom object.
- Where-used of extension fields now show the mashups created using Fiori adaptation.

5 Getting Started

5.1 Getting Started with the Studio

Getting Started encompasses the following reference material:

First Steps

Learn about the following:

- Installing SAP Cloud Applications Studio [page 30]
  You have to install the latest version of the SAP Cloud Applications Studio ("studio") from the SAP Service Marketplace. This document takes you through the requirements and steps to install the studio

  > **i Note**
  >
  > Prior to working in the studio, you must change the initial password provided to you in the SAP cloud solution.

- User Setup Quick Guide [page 33]
  Before you can begin working in the studio, the initial user has to create the required users for the studio and for the SAP cloud solution. This quick guide explains the entire process from creating service agents, requesting users for these service agents, assigning an ID and an initial password to each user, and assigning the required work centers. In addition, it also explains the process to create business users for testing purposes in the SAP cloud solution.
- User Roles [page 32]
  Find out about the different user roles in the studio:
  - For development purposes: Studio Administrator and Developer
  - For testing purposes: Business User
- **Features of the Studio**
  The studio enables you to adapt and enhance the solution capabilities of the SAP cloud solution. Using the studio, you can develop, deploy, and test specific add-on functionality.
  For an overview of the studio features, see About SAP Cloud Applications Studio [page 25].

  > **⚠ Caution**
  >
  > This documentation covers the studio as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every SAP cloud solution. Before using a particular feature, you should consult the Feature Matrix to determine whether the feature is available for the SAP cloud solution for which you are creating a solution. If the feature is available, it is marked with an X in the relevant column of the matrix.
  >
  > To check which features are available for which SAP cloud solution, see Feature Matrix [page 27].

- Localization Toolkit (SAP Business ByDesign only) [page 139]
  The localization toolkit is a set of development tools and instructions that enable SAP partners to extend and adapt the capabilities of the SAP Business ByDesign solution to provide a locally-compliant solution for customers.

Get an Introduction to Business Object Modeling and the Public Solution Model

- Introduction to Business Object Modeling [page 81]
  the SAP cloud solutions are built using the concept of business objects to model the business environment.
- Public Solution Model [page 77]
  The SAP public solution model (PSM) contains all entities in the SAP cloud solution that are released for use by external consumers. External consumers can be partners who develop solutions on top of the SAP cloud solution, such as add-ons and integration scenarios, or administrators in the SAP solution, who, for example, use SAP data sources to create new reports.
- Repository Explorer [page 49]
  The *Repository Explorer* tool window displays SAP content that is released with the public solution model (PSM) such as business objects, data types, enhancement options, inbound service interfaces, and reuse UIs. You can explore the content to gain an understanding of the repository structure and the business context and use of SAP content, which you can reuse and enhance in your solution.

Solution Examples

This documentation contains various solution examples. They include coding and show you how to use particular functions of the studio. You also learn how to create a complete solution in the studio, configure the screens in the UI designer and how to use the functions in SAP's cloud solution.

Build a solution to manage the allocation of car parking spaces to employees:

- Example: Building a Solution [page 99]

Try out particular functions of the studio:

-
- Example: Create a Code List Data Type (Purchase Order) [page 473]
- Example: Enable Notification Rules (Price Request) [page 504]
- Example: Changing the Status of a Price Request (Mass Data Run) [page 546]
- Example: Calculate Duration of Campaign (References to Customer-Specific Fields) [page 455]
- Example: Create an Approval Process (Price Request) [page 487]
- Example: Consume a REST Service [page 616]

> ⚠ Caution
>
> Any software coding or code lines/strings ("Code") provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio ("studio") as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

5.1.1 Update Your Language Preference

As a development user, you may have to update your default language to English.

To do this, follow these steps:

1. Log on to the **SAP Cloud for Customer** or **Business ByDesign** application with your development user.
2. From the user profile menu, navigate to *Settings* > *My Settings*.
3. In the *Language* field, select *English* from the dropdown menu.
4. Save your setting.
5. Log off and then log on again for your changes to take effect.

5.2 About the SAP Cloud Applications Studio

The SAP Cloud Applications Studio ("studio") for the SAP cloud solutions enables SAP partners, resellers, and customers to adapt and enhance the solution capabilities of the SAP cloud solutions.

The development tools provided are presented in an integrated development environment (IDE) based on Microsoft Visual Studio. Using the studio, you can develop, deploy, and test specific add-on functionality. Furthermore, you can create and integrate new business content, services, and user interfaces to provide complete micro-vertical business solutions.

The studio allows SAP partners, resellers, and customers to work with the same entities that SAP uses in the core development of its cloud solutions and to develop solution capabilities that have the same look and feel as the SAP standard cloud solutions.

The development of a customer-specific solution is triggered by a customer requirement. The solution is typically developed by a reseller for a specific customer.

For information about the development scenarios for customer-specific solutions, see Lifecycle Management of Customer-Specific Solutions [page 143].

Features

Integrated Development Environment

The studio is based on a local integrated development environment (IDE) that:

- Provides access to a hosted repository.
- Provides access to all tools you need to create and enhance the functionality of the SAP standard cloud solution.
- Is integrated with the tools that administrators work with in the SAP cloud solution.
- Allows you to manage the entire the entire lifecycle of customer-specific solutions, including development, testing, and assembly.
- Makes it easy and quick for you to perform activities through features such as wizards for multistep tasks.

- Simplifies programming through features such as code completion.

For information about the specific features and tools provided by the studio, see Working in the Integrated Development Environment [page 42].

Lightweight Scripting Language

You use the scripting language to define business objects and business object extensions and to implement the business logic for solution capabilities. The scripting language is a lightweight scripting language. Its syntax is easy to learn and to enter using a simple code editor.

For information, see Scripting Language Reference [page 189].

Model-Based Development

The SAP public solution model (PSM) contains all entities in the SAP cloud solution that are released for use by external consumers. External consumers can be partners who develop solutions such as add-ons and integration scenarios on top of the SAP cloud solution or administrators who, for example, use SAP data sources to create new reports in SAP' cloud solution.

For information about the basic principles of the PSM and SAP content types, see About the Public Solution Model (PSM) [page 77].

Designer for Desktop and Mobile User Interfaces

You use the UI designer to create new and replace existing user interfaces (UI). You can create and modify UI components using floorplans, patterns, and UI controls.

For information about the UI designer, see Introduction to the User Interface Designer [page 662].

Integration with Services

You can generate web services to access SAP business objects and the business objects of add-on solutions. You can also set up communication between business objects residing in different deployment units and import external XML data into business objects.

For information about web services, see Web Services Quick Guide [page 586].

You can also integrate data from the SAP cloud solution with data provided by an online web service or application by creating mashups. Mashups can include web searches, company or industry-specific business information or online map searches.

For information about mashups, see Mashups Quick Guide [page 551].

Translation Support

You can extract text from certain project items and export it in an XML-based file format for translation into one of the languages supported by the SAP standard cloud solution.

For information about the functions that support translation of a solution, see Translation Quick Guide [page 651].

End-to-End Software Lifecycle Management

Integration with the SAP cloud solution provides a strong lifecycle management backbone that supports the full life cycle of solutions.

For information about the development scenarios for customer-specific solutions, see Lifecycle Management of Customer-Specific Solutions [page 143].

Cloud Readiness

The studio supports the development of solution capabilities that are tailored for specific customers. The development and deployment of these solution capabilities causes no disruption to daily business because the upload process of a solution to a customer's production system does not require any system downtime.

5.3 Feature Matrix

This documentation covers the SAP Cloud Applications Studio ("studio") as a whole; therefore it describes all features that are available within the studio. Not all of these features are available for every SAP cloud solution. Before using a particular feature, please consult the matrix below to determine whether the feature is available for the SAP cloud solution for which you are creating a solution. If the feature is available, it is marked with **`Yes`** in the relevant column.

General Features

	SAP Business By-Design	SAP Hybris Cloud for Customer	SAP Cloud for Travel and Expense
Business Objects	Yes	Yes	Yes
Query	Yes	Yes	Yes
Print Forms	Yes	Yes	No
Translation	Yes	Yes	Yes

	SAP Business By-Design	SAP Hybris Cloud for Customer	SAP Cloud for Travel and Expense
Analytics	Yes	Yes	Yes
Approval Process	Yes	Yes	Yes
Notification Rules	No	Yes	No
Code List Data Type	Yes	Yes	Yes
Mass Data Run	Yes	Yes	Yes

Scripting Language

	SAP Business By-Design	SAP Hybris Cloud for Customer	SAP Cloud for Travel and Expense
Business objects • `Activity Task` • `Appointment Activity` • `Email Activity` • `Phone Call Activity`	Business objects • `Activity Task` • `Appointment Activity` • `Email Activity` • `Phone Call Activity`	`Activity` business object plus type code	Business objects • `Activity Task` • `Appointment Activity` • `Email Activity` • `Phone Call Activity`

Business Configuration

	SAP Business By-Design	SAP Hybris Cloud for Customer	SAP Cloud for Travel and Expense
BAC Elements	Yes	Yes	Yes
Business Configuration Object	Yes	Yes	Yes
Business Configuration Set	Yes	Yes	Yes
Business Configuration View	Yes	Yes	Yes
Fine-Tuning Project Template	Yes	No	No
Tax Decision Tree	Yes	No	No

Extension

	SAP Business By-Design	SAP Hybris Cloud for Customer	SAP Cloud for Travel and Expense
Business Object Extensions	Yes	Yes	Yes

	SAP Business By-Design	SAP Hybris Cloud for Customer	SAP Cloud for Travel and Expense
Enhancement Implementations	Yes	Yes	Yes
Embedded Components	Yes	Yes	Yes
Process Extension Scenarios	Yes	Yes	Yes
References to Customer-Specific Fields	Yes	Yes	Yes

> **i Note**
>
> For a list of supported Business Object Extensions, check the documents below:
>
> - Node (Business Object Extension) [page 227]
> - Action (Business Object Extension) [page 220]

Mashups

	SAP Business By-Design	SAP Hybris Cloud for Customer	SAP Cloud for Travel and Expense
All types of mashups	Yes	Yes	Yes

Screens

	SAP Business By-Design	SAP Hybris Cloud for Customer	SAP Cloud for Travel and Expense
Thing Inspector, Quick Create, Quick View	No	Yes	Yes
All other screen types	Yes	Yes	Yes
Object-Based Navigation	Yes	Yes	Yes
Thing-Based Navigation	No	Yes	Yes
UI Switch	Yes	Yes	Yes

Service Integration

	SAP Business By-Design	SAP Hybris Cloud for Customer	SAP Cloud for Travel and Expense
Communication Scenario	Yes	Yes	Yes
External Web Service Integration	Yes	Yes	Yes
Internal Communication	Yes	Yes	Yes
Web Service	Yes	Yes	Yes

	SAP Business By-Design	SAP Hybris Cloud for Customer	SAP Cloud for Travel and Expense
XML File Input	Yes	Yes	Yes

5.4 First Steps

5.4.1 Installing and Updating the SAP Cloud Applications Studio

Installing the Studio

To build solutions for the SAP cloud solutions install the SAP Cloud Applications Studio.

Prerequisites

- To download the studio from the SAP Software Download Center, you need a valid S user.
- You are using one of the following operating systems:
  - Microsoft Windows XP
  - Microsoft Windows Vista
  - Microsoft Windows 7 or higher
- You have installed the following:
  - Relevant version of Microsoft .NET Framework
  - Relevant version of Microsoft Visual Studio 2010 Shell (Isolated)
  - Relevant version of Microsoft Silverlight

> **i Note**
>
> Partners are required and solely responsible to regularly install security updates or patches provided by Microsoft Corporation for the Microsoft .NET Framework and Microsoft Visual Studio 2010 Shell (Isolated). As these components need to be installed/obtained by the partner, SAP is not responsible for installation of any related security updates or patches. For more information, see http://www.microsoft.com .

- The studio version must match the release version of the SAP cloud solution for which you are creating a solution.

Example

Release of the SAP Cloud Solution	Version of the Studio
SAP Business ByDesign **1702** SAP Cloud for Customer **1702**	SAP Cloud Applications Studio **1702**
SAP Business ByDesign **1711** SAP Cloud for Customer **1711**	SAP Cloud Applications Studio **1711**

To determine the release version in the SAP cloud solution, click *Help* > *About [SAP cloud solution]* and verify that the *Technical Release* details match the latest build version of the studio.

- **Previous Build**
  If relevant, you have removed the previous build. Check the sections below on updating your studio installation
- You have signed the relevant contract and you have an S-User for the *SAP Software Download Center* and for the Business Center for Cloud Solutions from SAP.
  If you do not have access or you do not see any search results, contact solution partner management.
- You have maintained your access details and those of any additional contacts in your company by using the *Manage My Partnership Application* at *www.sappartneredge.com* > *Partner Login* > *My Partnership* > *Manage My Partnership Application*.

Steps

1. Go to the *SAP Software Download Center* at https://support.sap.com/swdc.
2. Click *Installations & Upgrades* > *Access downloads*. The system opens the *Software Downloads* window.
3. In the *Search* field, enter **Studio XXXX** where **XXXX** stands for the studio release version. The system gives you a list of the available studio versions. .
4. Click the required SAP CLOUD APP STUDIO product version. The system takes you to your company's download catalog.

   Note

   If the search does not produce any results, you might not have a valid S user. In such a case, please contact solution partner management.

5. Click the *Downloads* tab, select your solution, and click the **arrow** icon. Choose the required solution and click *Download Basket* from the list of available actions.

   Note

   If you encounter any issues, contact SAP Partner Management by writing an e-mail to **CloudSolutionPartner@sap.com**.

Updating Your Studio Installation

Once you have installed the studio, you should make sure that you keep your studio installation up-to-date. SAP usually publishes new studio builds every two weeks in the SAP Software Download Center. You can check the build date of your installation under *Help* > *About SAP Cloud Applications Studio*.

Note that as of version 1302 of the studio you can use the *Check for Updates* function in the *Help* menu. When you select this function, the studio checks if a newer version of your studio is available on the *SAP Software Download Center* . In case a new version is available, you are informed accordingly and you can directly start an update process. The update process automatically removes the studio, downloads the latest version from the *SAP Software Download Center* and installs it. Ensure that you save your work before updating the studio. You require the same authorization for this process as for the manual download of the studio from the *SAP Software Download Center* . If you experience problems with the automatic download,check if you can manually download the studio from the *SAP Software Download Center* .

Automatic Check for Updates

You can enable the studio to check for updates automatically. To do this, in the studio, go to *Administration* > *Options and Settings* > *Update*, click the checkbox and enter how often the system should check for updates.

Automatic Update of Local Help

When you install a new build of the studio the local help is now also installed automatically.

5.4.2 User Roles

The studio has the following user roles:

Developer

Developers can perform the following tasks:

- Create and maintain all types of development objects, such as business objects and user interfaces, in the studio, user interface (UI) designer, and administrator tools for the SAP cloud solutions
- Access work center views for Analytics

For more information about what the developer role can do, see the reference documentation in the section Developer Desktop [page 360].

Studio Administrator

> ⚠ Caution
>
> The studio administrator must not be confused with the administrator in the SAP cloud solution. These users have different roles.

Studio administrators can perform all tasks that a developer can perform. In addition, studio administrators can perform the following tasks:

- Create new solutions
- Delete solutions
- Submit solutions for certification
- Submit patch requests
- Delete sessions and locks for other users

For more information about the tasks that only a studio administrator can perform, see:

- Administration Quick Guide [page 159]
- Implementation Manager Quick Guide [page 162]

Business User

The business user role is used for testing within the development or test tenant of the SAP cloud solution. Business users have authorizations and restrictions equivalent to users on a customer tenant; they have no authorization within the studio.

Business Users can perform the following tasks (depending on their access rights):

- Create additional business users
- Assign work center views to other business users
- Grant instance-based authorizations to other business users
- Perform business configuration scoping and deployment
- Start a trace from within the SAP cloud solution

Setting Up the Users

For information on how to set up the users for each of the roles, see User Setup Quick Guide [page 33].

5.4.3 User Setup Quick Guide

Before you can begin administrating, developing, or testing solutions in the SAP Cloud Applications Studio (“studio”), you have to create development users (studio administrator and developer user roles) and business users. You create these users in the SAP cloud solution

After these users have been created, the development users must change their initial passwords and maintain contact data. Business users only need to change their initial passwords.

> ⚠ Caution
>
> Development users with studio administrator or developer user roles are for use in the studio only.
>
> Business users are for use in the SAP cloud solution.

Business and Technical Background

User Roles

The user roles are Studio Administrator, Developer, and Business User.

For more information, see User Roles [page 32].

Administration Toolbar and Implementation Manager

The studio administrator user has authorization for the *Administration* toolbar and the *Implementation Manager*, which provides access to a number of administrative tasks.

For more information, see:

- Administration Quick Guide [page 159]
- Implementation Manager Quick Guide [page 162]

User and Access Management in the SAP Cloud Solution

To manage users and assign access rights, choose:

SAP Cloud Solution	Work Center View
SAP Business ByDesign	*Application and User Management* > *User and Access Management* > *Business Users*
SAP Cloud for Customer	*Administrator* > *General Settings* > *Users* > *Business Users* or *Application and User Management* > *Business Users*

SAP Cloud Solution	Work Center View
SAP Cloud for Travel and Expense	*Application and User Management* > *User and Access Management* > *Business Users*

Tasks

Prerequisites

- Prerequisites for creating service agents, assigning user IDs and passwords, assigning work centers, and creating business users:
  - You have an initial business user. The initial user is provided to you during the tenant provisioning process. Only use the initial business user in the SAP cloud solution. Do **not** use it in the studio.
  - To create the service agent, you need access to the following work center views:

SAP Cloud Solution	Work Center View
SAP Business ByDesign	*Business Partner Data* > *Service Agents* *Project Management* > *Service Agents* *Resource Management*
SAP Cloud for Customer	*Business Partners* > *Service Agents*
SAP Cloud for Travel and Expense	*Business Partners* > *Service Agents*

> **i** Note
>
> For SAP Cloud for Travel and Expense:
>
> To ensure you have access to the *Business Partners* work center, follow the instructions related to the initial user in the **Master Guide for SAP Cloud for Travel and Expense** in the sections **Define Project Scope** and **Create Service Agent**.
>
> Before you start working with the Master Guide, make sure you have the latest version. You can find the latest version at the following location: http://service.sap.com/instguides under *Cloud Solutions from SAP* > *SAP Cloud for Travel and Expense*.

- To assign access rights to the business user, you need access to the following work center views:

SAP Cloud Solution	Work Center View
SAP Business ByDesign	*Application and User Management* > *Business Users*
SAP Cloud for Customer	*Administrator* > *General Settings* > *Users* > *Business Users* or *Application and User Management* > *Business Users*
SAP Cloud for Travel and Expense	*Application and User Management* > *Business Users*

- Prerequisites for maintaining contact data:
  You have a service agent user ID that matches the technical ID of the service agent.

Tasks of the Initial User

Before starting to develop in the studio, you have to create the required development users with the relevant administration or developer rights. You do this by creating service agents in the SAP cloud solution, and assigning the relevant user ID, password, and work centers to those service agents.

Create a Service Agent and Request a User (for Development Users)

1. Open the *Service Agents* view as follows:

SAP Cloud Solution	Work Center / Work Center View
SAP Business ByDesign	Business Partner Data —> New Service Agent Project Management —> New Service Agent Resource Management —> New Service Agent
SAP Cloud for Customer	Business Partners —> New Service Agent Project Management —> New Service Agent Resource Management —> New Service Agent
SAP Cloud for Travel and Expense	Business Partners —> New Service Agent

2. Click *New* and select *Service Agent*.
   The *New Service Agent* quick activity opens.
3. Enter a service agent ID and a name.

> ⚠ Caution
>
> If the *Service Agent ID* field is enabled, your entry must have the prefix **`EXT`**.
>
> If internal number assignment is active, the system determines the new ID.
>
> For the ID, the system uses the naming convention <LastName><FirstName><ServiceAgentID>. For example, the service agent ID for Kate Jacob could be `JACOBKATEEXT999`.

4. Note down the service agent ID.
5. Save the service agent and click *Request User*.
   The system creates a user for the service agent.
6. Click *Save* and close the editor.

Assign a User ID and a Password to the Service Agent (for Development Users)

1. Open the *Business Users* view by clicking *Application and User Management* > *User and Access Management* > *Business Users*.
2. Search for the service agent (business user) that you created. For the search, enter one of the following:
   - *<ServiceAgentID> (for example, `*EXT999`)
   - Name
3. Click *Edit* and select *Attributes*.
   The *Business User* quick activity opens.
4. In the *User Data* screen area, define a user ID, for example, the last name of the service agent.
5. In the *Language* field, set the language to **`English`**.
   You must set the language to English to ensure the consistency of your solution, because the logon language of both the SAP cloud solution and the studio is English
6. In the fields *Password* and *Confirm Password*, assign an initial password.
7. Click *Save* or *Save and Close*.

Assign Work Centers to the Service Agent (for Development Users)

1. Open the *Business Users* view as follows:

SAP Cloud Solution	Work Center View
SAP Business ByDesign	*Application and User Management* > *Business Users*

SAP Cloud Solution	Work Center View
SAP Cloud for Customer	*Administrator* > *General Settings* > *Users* > *Business Users* or *Application and User Management* > *Business Users*
SAP Cloud for Travel and Expense	*Application and User Management* > *Business Users*

2. Search for the service agent (business user) that you created. For the search, enter one of the following:
   - *<ServiceAgentID> (for example, `*EXT999`)
   - Name
3. Click *Edit* and select *Access Rights*.
   The *Edit Access Rights* quick activity opens.
4. In the *Edit Access Rights* editor, on the *Work Center and View Assignment* tab assign the following work centers and views by selecting *Assigned to User*.

> ⚠ Caution
>
> The work centers and views that are available for assignment depend on the SAP cloud solution, therefore not all work center views may be available to you.
>
> For more information about the features that are available to you in the studio and for some of which you require these access rights, see Feature Matrix [page 27].

> → Tip
>
> You can also use business roles to assign access rights to users. In the *Edit Access Rights* editor, click the *Business Roles* tab. For more information, see the Business Roles Quick Guide in your SAP cloud solution.

Work Center ID / Work Center Name (in Available Work Centers)	Work Center View ID / Name (in Available Views in Work Center: <Name of Work Center>
PDI_PARTNER_DEVELOMPENT / Partner Development	○ **Developer User Role** Select the following views: ○ PDI_DEVELOPMENT / Development ○ **Studio Administrator User Role** Select **all** of the following views: ○ PDI_ADMINISTRATION / Administration ○ PDI_DEVELOPMENT / Development **i Note** You need at least one user with a studio administrator role For more information, see User Roles [page 32]
ANA_BUSINESSANALYTICS / Business Analytics	Select **all** of the following views, if available: ○ ANA_DESIGNREPORTS / Design Reports ○ ANA_DESIGN_DSO / Design Data Sources ○ ANA_KUA_KF_OWL_WOC_VIEW / Design Key Figures
○ In SAP Cloud for Customer: Administrator or TS_APPLICATIONUSERMANAGEMENT / Application and User Management ○ In all other SAP cloud solutions: ITS_APPLICATIONUSERMANAGEMENT / Application and User Management	Select **all** of the following views, if available: ○ APPROVALPROCESSES_WCVIEW / Approval Processes ○ ITS_FORMTEMPLATEMAINT / Form Template Maintenance ○ ITS_MASTERTEMPLATEMAINT / Master Template Maintenance ○ ITS_INCIDENTS / Incidents ○ ITS_OFFICETEMPLATEMAINT / Microsoft Office® Template Maintenance ○ ITS_KEYUSERCHANGELOG / Flexibility Change Log ○ MAS_MASHUPAUTHORING / Mashup Authoring ○ MAS_WEBSERVICEAUTHORING / Mashup Web Services ○ NOTIFICATION_DEFINITION / Notification Rules

Work Center and View Assignment

5. Click *Save* or *Save and Close*.
6. Notify the users that you have created a user ID and an initial password.

⚠ Caution

The users must change the initial password. To do this, they must log on for the first time to the SAP cloud solution using the assigned user ID. The system requests a new password.

The user can use the same password in both the SAP cloud solution and in the studio.

In addition to the users for the studio, you also have to create business users for use in the SAP cloud solution only. These business users are for test purposes and are not for use in the studio.

Create and Assign Business Users (Test Users)

The Business User role is used for testing within development tenant or test tenant of the SAP cloud solution. Business users have authorizations and restrictions equivalent to users on a customer tenant.

> ⚠ Caution
>
> Do **not** assign the *Partner Development* work center and its views to a business user role. These users must not have access to the studio.

You use business users to perform the following in the SAP cloud solution:

- Scoping and deployment of your business configuration entities
- Testing of your solution
- Starting a trace from within the SAP cloud solution

Steps

1. Create the required business users (administrator users and end users) in the SAP cloud solution.
2. Assign the required work centers and work center views to the business users.

   > ℹ Note
   >
   > To enable testing of business configuration entities, at least one standard business user must be assigned to the *Business Configuration* work center.
   >
   > Do not test or change business configuration entities with a development user.
3. Grant instance-based authorizations to the business users.

Development Users

Change Your Initial Password

> ⚠ Caution
>
> After receiving your user details, **before** logging on to the studio, you must first change your initial password in the SAP cloud solution. This is important, because you cannot change the initial password in the studio.

1. Launch the SAP cloud solution.
2. Enter your user ID and the initial password that has been created for you by the initial user.
   The system requests a new password.
3. Enter a new password and confirm the entry.
   You can now also log on to the studio with your user ID and the new password.

> ⚠ Caution
>
> Do not change the logon language defined for your user ID, as this causes inconsistencies in the solution. The logon language must be English.

Maintain Your Contact Details

As a development user, you may have to create incidents for SAP Support. To ensure that these incidents are sent directly to SAP Support, you must maintain your telephone number and e-mail address.

For information about reporting an incident in the SAP cloud solution, see Report an Incident [page 628].

> i Note
>
> If you do not maintain your contact details, when you create an incident the system forwards the incident automatically to a development user in your organization, who has maintained contact data.

1. Log on to the SAP cloud solution with your development user and proceed as follows:

SAP Cloud Solution	Work Center View
SAP Business ByDesign	*Home* > *Self-Services* > *Self-Services Overview* > *Company Address Book* > *Edit My Contact Data*
SAP Cloud for Customer	*[Your User Name]* > *Profile*
SAP Cloud for Travel and Expense	*Personalize* > *My Data*

2. Enter your telephone number and e-mail address and save your entries.

Business Users

Change Your Initial Password

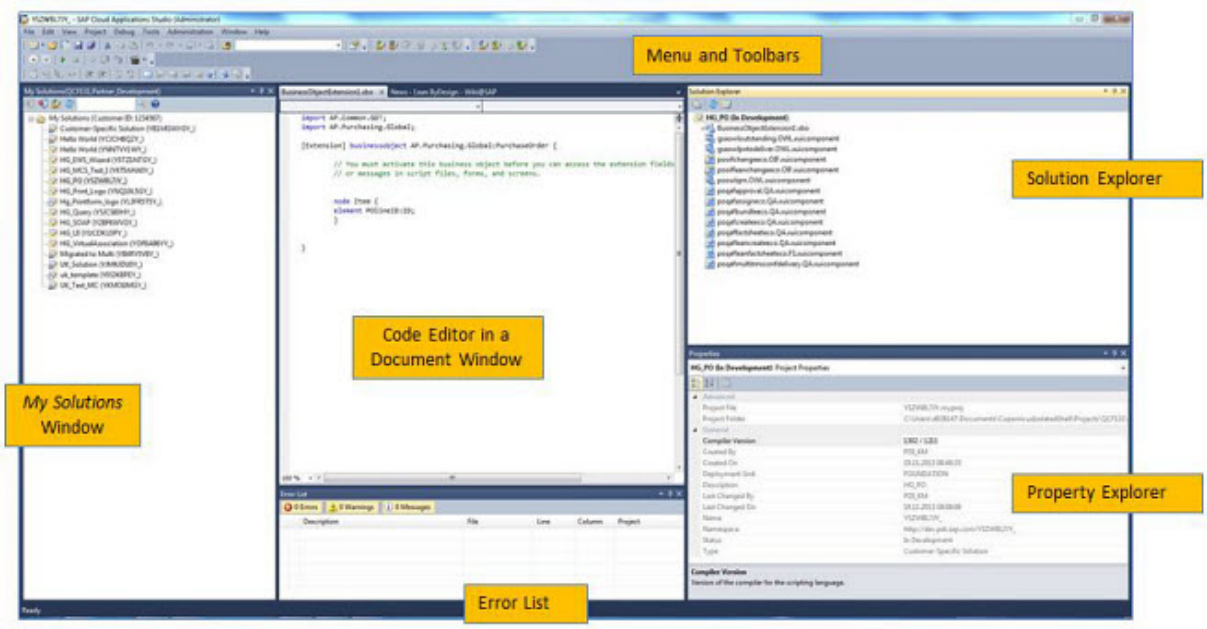
> ⚠ Caution
>
> After receiving your user details, you must first change your initial password in the SAP cloud solution.

1. Launch the SAP cloud solution.
2. Enter your user ID and the initial password that has been created for you by the initial user.
   The system requests a new password.
3. Enter a new password and confirm the entry.

5.5 Introduction to the Integrated Development Environment

5.5.1 Working in the Integrated Development Environment

The development tools provided by the release are presented in an integrated development environment (IDE) based on Microsoft Visual Studio. The user interface (UI) of the release, referred to as the developer desktop, is composed of different elements. Many elements are standard features and tools of Microsoft Visual Studio, for example, the *Menu* toolbar, the *Standard* toolbar, various tool windows, and the document window. In addition, the developer desktop provides UI elements that represent specific features and tools of the release development environment. To get an idea of the integrated development environment, have a look at the following screenshot:



What the integrated development environment looks like

The UI elements of the developer desktop are docked or auto-hidden on the left, bottom, and right. The tool windows, menus, and toolbars available depend on the type of solution or item you are working on as well as on the settings you have applied. You can move and position the UI elements to suit your working behavior.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

You can access the release documentation from the *Help* menu by selecting *SAP Cloud Applications Studio Help*. The library documentation appears in your browser.

For more information, see .

> **i** Note
>
> This documentation does not describe the standard features and technologies in Microsoft Visual Studio. For information about Microsoft Visual Studio, see http://www.microsoft.com .

Developer Desktop

The following sections describe the tools that you can use for the administration and lifecycle management of a solution and how you can access the corresponding user interface (UI) elements.

Administration toolbar

The *Administration* toolbar provides a number of administrative tasks, such as creating and deleting customer-specific solutions or solution templates and managing user sessions. You can also access the *Session Administration* tool window from the *Administration* toolbar.

You can access the *Administration* toolbar in the *View* menu by clicking *Toolbars* *Administration*.

For more information, see Administration Quick Guide [page 159].

Session Administration

The *Session Administration* tool window displays a list of users in the release with open sessions. If you are assigned the Administrator role, you can delete user sessions and any item locks that are applied for the corresponding sessions. All other users can delete their own sessions and item locks, but they cannot delete other users' sessions and item locks.

You can access the *Session Administration* tool window in the *View* menu and in the *Administration* menu.

Implementation Manager

The *Implementation Manager* (). is a tool for managing the lifecycle of customer-specific solutions. You can access the lifecycle management functions provided by the release from a toolbar and a tool window:

- *Implementation Manager* tool window
  The tool window provides a view of the solution status and implementation log at each stage in the process.

You can access the *Implementation Manager* tool window in the *View* menu and in the *Administration* menu.

- *Implementation Manager* toolbar
  The toolbar provides buttons that allow you to assemble, upload, and create a patch for your solution.
  You can access the *Implementation Manager* toolbar in the *View* menu by clicking *Toolbars* > *Implementation Manager*.

For more information, see Implementation Manager Quick Guide [page 162].

Administration menu

You can access existing administration and lifecycle management functions from the *Administration* menu. The *Administration* menu also gives you access to most of the functions provided by the *My Solutions* tool window.

The functions available on the *Administration* menu are:

- *Log On*
  For information, see Log on to the Repository [page 19].
- *Log Off*
- *Create Solution*
  For information, see Create a Customer-Specific Solution [page 177].
- *Delete Solution*
- *Session Administration*
- *Solution Administration*
  Here you can access the following functions: *Trigger Test Deployment*, *Request Quality Review*, and *Create Patch*.
- *Implementation Manager*
- *Admin Log*
- *Switch Customer*
  For information, see Switch a Customer Assignment [page 158].
- *Open UI Designer*
  If you have opened a solution, you can navigate to the UI designer by using this function.
  For more information, see Introduction to the User Interface Designer [page 662].
- *Switch On/Off Admin Mode in SAP Solution*
  If you want to define analytics content or create a mashup, you must switch on the admin mode in the studio. This makes sure that the system saves your content and any changes you make in your solution.
  Here is a list of the functions that use the admin mode:
  - Mashups (Data, HTML, URL), see Creating Mashups [page 553]
  - Mashup Web Services (SOAP, REST, RSS/Atom), see Create a Mashup Web Service [page 559]
  - Combined Data Sources, see Create a Combined Data Source [page 530]
  - Joined Data Sources, see Create a Joined Data Source [page 532]
  - Create a Key Figure, see Create a Key Figure [page 535].
  - Reports, see Create a Report [page 534]

  > **i Note**
  >
  > You cannot switch on the admin mode for solution templates. For more information, see Solution Templates [page 174].

- *Log On to System via Browser*
  You can open the SAP cloud solution from the studio by selecting this option.
- *Update Authorizations and Access Rights*
  For more information about when to use this function, see Assign Access Rights [page 427].
- *Options and Settings*

The following sections describe the tools that you can use to develop a solution and how you can access the corresponding user interface (UI) elements.

My Solutions

The *My Solutions* tool window provides functions that allow you to create solutions and to access the repository. You must log on to a repository system to create and edit solutions in the studio.

For more information, see My Solutions [page 48].

Repository Explorer

The *Repository Explorer* tool window displays SAP content that is released with the public solution model (PSM) such as business objects, data types, enhancement options, inbound service interfaces, and reuse UIs. You can explore the content to gain an understanding of the repository structure and the business context and use of SAP content, which you can reuse and enhance in your solution.

For more information, see Repository Explorer [page 49].

Solution Explorer

The *Solution Explorer* tool window displays a solution and the project items created for this solution, for example, a business object or a screen. In the *Solution Explorer*, you can open items for editing and add new items to a solution. You can also search for files within a solution.

For more information, see Solution Explorer [page 54].

Add New Item dialog box

The *Add New Item* dialog box displays the installed item templates that SAP provides for solution development. You can add items such as business objects, different user interface (UI) components, and Web services to your solution.

For more information, see Add New Item Dialog Box [page 58].

Properties window

The *Properties* window displays information about the item in your solution that is currently selected in the *Solution Explorer*. When you select the solution, you can view solution properties such as the solution status. When you select an item such as a business object or a UI component, you can view item properties such as the activation status of runtime objects.

For more information, see Properties Window [page 67].

Options dialog box

You can use the *Options* dialog box to configure the studio to meet your needs. You can view or define settings for the code editor as well as those relating to connectivity, debugging, runtime metadata, external tools, debugging and tracing, and update.

You can access the *Options* dialog box in one of the following ways:

- On the *Connect to Repository* screen, click the pencil icon ( ) to the right of the *System* field.
- In the *Administration* menu, click *Options and Settings*.

You can change your settings for the following options:

Text Editor

Define the default behavior of the code editor for business objects, business object extensions, and actions and events. For example, you can display line numbers in the code editor for reference purposes. To access these settings, in the navigation pane of the *Options* dialog box, expand the *Text Editor* node and then expand the subnode for your project item.

SAP

Click the tabs on the right to access the following options:

- *Connectivity*: Enter the name, host, and proxy settings of your repository system. You can also enter your user and password and set a default system. Click *Save* to save your settings.
- *Runtime Metadata*: Clear the cache.
- *Debugging and Tracing*: Enter your debugging and tracing options.
  For more information, see About Debugging [page 631].
- *Update*: Select the *Check for Updates* checkbox. Next time you log on to the studio, if a new studio build is available, the system prompts you to confirm that the system should be updated.
  For more information, see Installing and Updating the SAP Cloud Applications Studio [page 30].

Document windows

In the release, document windows provide a view of the items that you add to your solution. Examples of items that you can open, edit, and save in a document window are business object definitions, extensions for existing business object definitions, data sources, and business object queries. The type of document window in which an item opens depends on the type of item. For example, when you open a business object definition, the business object file is displayed in the code editor.

Document windows are displayed on tabs in the center of the developer desktop. You can switch between open document windows by using the tabs. You can also view multiple document windows side-by-side.

You can open an item in a document window by double-clicking the item in the *Solution Explorer*.

> **i** Note
>
> If you double-click an item that represents a user interface component, such as a quick activity floorplan (QAF) screen or a business configuration view, the item file opens in a separate window in the UI designer.

Error List window

The *Error List* displays errors, warnings, and messages produced as you edit code, such as syntax errors.

You can display the *Error List* in the *View* menu by selecting *Error List* and then use the *Errors*, *Warnings*, and *Messages* buttons to select the entries you want to display. By default, the *Error List* is displayed at the bottom of the developer desktop below the document windows.

Output window

The *Output* window provides several views depending on the mode you are working in. The following views are available:

- *Solution*
  If you are in design mode, this view displays status messages for business objects. These messages include errors that occur when you activate, check or clean a solution or a business object. For example, the *Output* window displays a message if it is not possible to compile a business object.
  For more information, see “How you can solve business object activation errors" in Activate a Business Object [page 420].
- *Debug*
  If you have started debugging, this view displays tracing information that the system is collecting during the debug run. For more information, see Debugging User Interface [page 635].

You can access the *Output* window in the *View* menu by clicking *Output*.

Wizards

The wizards in the studio guide you through a logical sequence of steps, making it easier and quicker for you to perform development activities that are complex, unfamiliar, or infrequently performed. All wizards are easy to access and follow similar user interface (UI) patterns.

For more information, see Working with the Wizards [page 70].

Debug user interface

The release's debugging user interface is composed of several elements, which enable you, for example, to choose debugging functions, to view information provided by the debugger, and to specify debug settings. For more information, see Debugging User Interface [page 635].

Trace Explorer

The *Trace Explorer* is a tool for gathering and analyzing diagnostic information.

For more information, see Trace Explorer [page 69].

5.5.2 My Solutions

The *My Solutions* tool window provides functions that allow you to create solutions and to access the repository. You must log on to a repository system to create and edit solutions in the studio.

Features

When you open the studio, you first log on to the repository. By default, the *My Solutions* tool window is displayed on the left of the developer desktop. You can also open the *My Solutions* window in the *View* menu.

The *My Solutions* window shows all solutions that are available in the repository. The solutions are displayed as follows: [Solution Type] [Solution Description] (Technical Name), for example: CarParkSolution (YQE1PZ2IY) or BCTest (YA2F975KJ).

You can access the following functions from the *My Solutions* tool window:

- *Log On*
  For information about logging on to the repository, see Log on to the Repository [page 19].
- *Create Solution*
  For more information about creating solutions, see the following documents:
  - Create a Customer-Specific Solution [page 177]
  - Create a Solution Template [page 175]
- *Refresh*
- *Search*
  You can search solutions by entering the name of the solution in the search field. If you do not know the exact name of an solution, you can just enter your search term and press *Enter*. The search is not case-sensitive. The first solution containing the search term is displayed at the top of the list. To get to the next hit, press *Enter*.
- *Log Off*
- *Delete Solution*
  To delete a solution, right-click the solution and select *Delete Solution*.

- *Open Solution*
  To open a solution, right-click the solution and select *Open Solution*. You can also open a solution by double-clicking it.
- *Solution Properties*
  If you want to display the solution properties, in the *My Solutions* window, right-click the solution and select *Solution Properties*. In the *Solution Properties* window, you can also add or change the contact information.

> → Tip
>
> You can also access some of these functions from the *Administration* menu.

Related Information

Solution Explorer [page 54]

5.5.3 Repository Explorer

The *Repository Explorer* tool window displays SAP content that is released with the public solution model (PSM) such as business objects, data types, enhancement options, inbound service interfaces, and reuse UIs. You can explore the content to gain an understanding of the repository structure and the business context and use of SAP content, which you can reuse and enhance in your solution.

Prerequisites

To display SAP content in the *Repository Explorer*, you must log on to the repository. For information, see Log on to the Repository [page 19].

Features

You can open the *Repository Explorer* tool window in the *View* menu by clicking *Repository Explorer*. You can dock the tool window on the edge of the developer desktop or as a tabbed document. For best readability, we recommend that you dock the *Repository Explorer* as a tabbed document.

The *Repository Explorer* displays different content types on separate tabs. The window is split into two panes: a tree view of the entities and a documentation pane. You can view the documentation for an entity by clicking the entity in the tree view. You can position the documentation pane at the bottom by clicking the *Horizontal* button or on the right by clicking the *Vertical* button.

Business Objects Tab

On the *Business Objects* tab, you can explore all SAP business objects that are released with the public solution model (PSM) by doing the following:

- **Explore the repository** — When you open the *Repository Explorer*, the tree view displays all SAP business objects. You can expand the tree view to display the business objects nodes, the node elements, and the relationship between business object nodes, which is called an association. You can also show and hide entities in the business object model by using the following buttons:
  - *Actions*
  - *Associations*
  - *Nodes*
  - *Queries*
    You can execute the query by selecting *Execute Query* in the context menu of the query. For more information, see Execute a Query [page 52].
- **Search business objects** — You can search for an SAP business object by entering the name of the business object in the search field. You can also filter the search results by namespace and/or deployment unit.

  > → Tip
  >
  > If the system cannot find any entities that match your search term, you can try the following:
  >
  > - Click the *Clear Search* button ( ) to search in all namespaces and development units (for business objects) or usage categories (for data types).
  > - Enter a different spelling or search term.

- **View information** — You can hover over an entity in the tree view to display information about the entity such as the object category or the subelement type. The information also indicates whether write access is allowed for an entity and whether the entity is deprecated.
- **View documentation** — You can click a business object and its elements to display comprehensive documentation in the documentation pane. On the *Description* tab, you see the business object (element) documentation; the data type information is displayed on the *Data Type* tab of the documentation pane and is also accessible via the *Data Types* tab as described below.

Data Types Tab

On the *Data Types* tab, you can explore all SAP data types that are released with the public solution model (PSM) by doing the following:

- **Search data types** — You can search for an SAP data type by entering the name of the data type in the search field. You can also filter the search results by namespace and/or usage category, for example, *Action*, *Key*, or *Message*.
- **View information and documentation** — You can hover over an entity in the tree view to display information about the entity such as the namespace. The information also indicates whether you can extend a data type and the usage category of the data type. The usage category describes in which context a data type can be used. For more information, see the following table.

Examples of Usage Categories for Data Types

Usage Category	Description
Action	Can only be used to define the parameter structure of an action.
Code Data Type Context	Can only be used as context structure of a code data type.
Core	A core data type according to CCTS that has unrestricted usage.
Enhancement Option	Can only be used in enhancement implementations.
Filter	Can only be used to define the filter structure of associations.
Key	Can be used as key in business object node structures, but not in messages.
Message	Can only be used to type messages
Message Intermediate	Can only be used in message types, but only at intermediate levels and not for the message type itself.
Message Unrestricted	Can be used in messages but not in business objects.
Unrestricted	Can be used in all data type elements.

Enhancement Options Tab

On the *Enhancement Options* tab, you can explore all SAP enhancement options that are released with the public solution model (PSM) by doing the following:

- **Search enhancement options** — You can search for an SAP enhancement option by entering the name of the enhancement option in the search field. You can also filter the search results by namespace.
- **View information and documentation** — You can hover over an entity in the tree view to display information about the entity such as the namespace. The information also indicates the usage type of the enhancement option (single use or multiple use) and the related business object.

  The documentation of the enhancement options describes how you can use an enhancement option, when the enhancement option is called, and what it does. Furthermore, it gives information about the business configuration content you need to develop for an enhancement implementation, the scoping requirements in the customer system, as well as information about the filters, the input parameters and output parameters, and testing activities.

Inbound Service Interfaces Tab

On the *Inbound Service Interfaces* tab, you can explore all SAP inbound service interfaces and service operation that are released with the public solution model (PSM) by doing the following:

- **Search inbound service interfaces** — You can search for an SAP inbound service interface by entering the name of the inbound service interface in the search field. You can expand the tree view to display the service operations. You can also filter the search results by namespace.
- **View information and documentation** — You can hover over an entity in the tree view to display information about the entity such as the namespace.

Reuse UIs Tab

On the *Reuse UIs* tab, you can explore the SAP reuse UIs that are released with the public solution model (PSM). The reuse UIs are grouped by UI type:

- Embedded Component
- Modal Dialog
- Object Instance Floorplan
- Object Value Selector
- Quick Activity Floorplan

You can do the following:

- **Search reuse UIs** — You can search for a reuse UI by entering a case-sensitive search term in the *Search* field.
- **View information** — You can click a reuse UI to display the `UI Type` and `Repository Path`.
- **Navigate to UI designer** — You can open the reuse UI by right-clicking the UI and selecting *Open in UI Designer*.

Related Information

About the Public Solution Model (PSM) [page 77]
Look Up the Representation Term of a Data Type [page 96]

5.5.4 Execute a Query to Explore a Business Object

The studio enables you to execute queries for SAP business objects and custom business objects to help you better understand the business object data model.

For custom business objects, you can execute the QueryByElements query, which is generated automatically, and complex custom queries. For more information, see Create a Query [page 421]

The following table shows you where to access the function:

SAP Business Object	Custom Business Object
Repository Explorer	*Solution Explorer*

> **i** Note
>
> You cannot execute a query on a customer's production tenant.

Procedure

1. **To execute a query for SAP business objects, proceed as follows:**
   1. In the studio, select *View* > *Repository Explorer.*
   2. Select the *Business Objects* tab and make sure that the *Queries* button is selected.
   3. Select a business object (or search one using the *Search* field) and expand the tree view.
   4. Select a query ( ), right-click it, and select *Execute Query*.
      The *Query Parameter Selection* window opens.

   To execute a query for custom business objects, proceed as follows:
   In the studio, in the *Solution Explorer*, right-click a business object or a query and select *Execute Query*. The *Query Parameter Selection* window opens.

   > **i** Note
   >
   > If your business object contains various queries, choose the query that you want to execute in the *Select Query* window.

2. Optional: In the *Query Parameter Selection* window, you can do the following:
   - *Add* or *Remove* parameters
   - *Paste Values from Clipboard* into the *Lower Boundary* field
   - Define the *Maximum Number of Records*
   - Calculate the *Available Number of Records for Selection*
3. To define the selection parameters, do the following:
   1. Click the *Parameter* field to select a parameter.
   2. To exclude records from the results, click the *Include/Exclude* field and select *Exclude*. The default value is *Include*.
   3. Click the *Option* field to select one of the following selection options: *Equal to*, *Between*, *Contains Pattern*, *Less Than*, *Less Than or Equal to*, *Greater Than*, *Greater Than or Equal to*.
   4. Enter a *Lower Boundary Value* and an *Upper Boundary Value*, if required.
4. Click *OK* to execute the query.

Result

The results list is displayed in the documentation pane. You can use the following options:

- *Navigate by Association*
  Explore the query results by following the associations of the business object. Please note that this is not possible, if the debugger is active.
- *Reopen Search*
  Click this button to reopen the *Query Parameter Selection* and adjust the selection parameters.
- *Single-Record View*
  Select a record and press the button to display the record in the *Single-Record View* window.
- *Search* window
  Search for a string or term in the query results.
- *Output Formatting Active*
  Toggle between the technical view and the user view. If you select *Output Formatting Active*, the user view is displayed.
- *Generate Code*
  Click this button to generate the `.absl` code that executes the query with the selection parameters used. You can then copy this code into a new or existing `.absl` file. In case there is already coding in your `.absl` file, there may be conflicts due to duplicate imports or variables, which you need to resolve.

> → Tip
>
> If you cannot see all the buttons, click the down arrow on the right or enlarge the query results window.

5.5.5 Solution Explorer

The *Solution Explorer* tool window displays a solution and the project items created for this solution, for example, a business object or a screen. In the *Solution Explorer*, you can open items for editing and add new items to a solution. You can also search for files within a solution.

When you double-click a solution in the *My Solutions* tool window, the solution opens in the *Solution Explorer*. By default, the *Solution Explorer* tool window is displayed on the right of the developer desktop. You can also open the *Solution Explorer* in the *View* menu by clicking *Solution Explorer*.

In the *Solution Explorer* window, the solution is displayed with its description (= the name you have given your solution) and the status information (for example, `In Development`). Below the solution, you can see all the items and item objects of the solution.

At the top of the *Solution Explorer* window, you can: view the properties of your solution and items, add a new item, refresh your solution, search for specific files within the solution, create correction and close correction during a production bug fix. The following table describes the icons that are available in the toolbar.

	View the properties of the solution and items		Refresh the solution
	Add a new item to the solution		Search for files in the solution
	Create a correction during production bug fix		Close a correction during production bug fix

The *Solution Explorer* uses icons in front of the items and item objects to display status information. Depending on the status, the icons used for items and objects change. For example, added to an item or object means "checked out by me" and means that the item has no up-to-date runtime objects. As an example, the following table shows the icons used for business objects:

	Runtime objects of business object are up-to-date		Business object has been checked out by me
	Runtime objects of business object are **not** up-to-date		Business object has been checked out by another user

The activation status of the runtime objects of an solution is displayed in the *Properties* window. For more information, see Properties Window [page 67].

Project

The project node displays the name of the project and the status of the solution, for example, *In Development*. When you create a solution for the first time, the status of the solution is *In Development* and you can use all available functions. When the status of the solution changes, for example, to *Assembled*, you can view the solution, but you cannot make any changes. For more information about the life cycle of a solution, see Lifecycle Management of Customer-Specific Solutions [page 143].

You can create folders to structure the items according to your preferences. Currently, items cannot be moved between folders in the *Solution Explorer*.

Generic Actions

Depending on the status of your solution, you can perform various actions for an item or the solution by right-clicking a node or item. Actions that you cannot choose for the solution or item you have selected are grayed out. Here is an overview of the actions that are available in the *Solution Explorer*:

- **Activate**
  You can activate an item or solution to generate the runtime objects that are required for an item or the solution to function. You must activate any new or modified items before you can test them in the SAP cloud solution.

- To activate all items of your solution, select *Activate* > *All Items*.
- To activate only the new or modified items, select *Activate* > *New and Changed Items*. This option speeds up the activation process.
- To determine if an item has been activated, in the *Solution Explorer*, select the item. The activation status is displayed in the *Properties*.

For more information on the activation of business objects, see Activate a Business Object [page 420].

- **Check**

  You can use the following check functions:
  - For a selected item of the solution:
    Right-click the element and select *Check*.
  - For all items of the solution:

    Right-click the project node and select *Check* > *Items*.
  - For all items and runtime objects of the solution:

    Right-click the project node and select *Check* > *Items and Runtime Objects*. This option checks the consistency of the design-time files and of the generated runtime objects.
  - For running consistency check in background mode:
    1. Right-click the project node and select *Check* > *Enable Background Mode*.
    2. Click on *Check* > *Items* or *Check* > *Items and Runtime Objects* to trigger the background job. When the mode is triggered, you can continue to work on other solutions, and avoid the timeout errors for large solutions.
    3. Click on *Check* > *Display Logs* to see the results of your check. The system shows the results in a pop-up window confirming if the consistency check is successful or not.

       Click on *Check* > *Disable Background Mode* to switch to the earlier context menu option that runs the check in fore-ground mode.

- **Check In/Check Out/Check In All Files/Revert**

  Items that you have checked out (for example, when changing them) are locked by you and other users cannot edit these items. A red check mark identifies items that you have checked out. When you have finished editing an item, you can check it in to make it available for other users to edit. You can also check in all items at the same time by right-clicking the solution and selecting *Check In All Files*. Items checked out by other users are identified by a blue user icon. The *Properties* window displays the user that has checked out the items.

  The .absl file that you have changed during production bug fix can be reverted to its previous checked in and activated version when you use the *Revert* action.

- **Add**

  You can access the following functions via the *Add* option:
  - *Add New Item*

    Opens the *Add New Item* dialog box that displays the installed item templates that SAP provides for solution development. For more information about the items you can add to your solution, see Add New Item Dialog Box [page 58].

    > → Tip
    >
    > You can also open the *Add New Item* dialog box by clicking the button at the top of the *Solution Explorer* tool window.
  - *Add Existing Item*

To document your solution, you can add file types such as `doc`, `ppt`, `xls`, `pdf`, `txt`, and `xlf` to your solution. In addition, you can copy files from other solutions to your solution, for example, a `.uicomponent` file.

- *Add New Folder*
  To organize the files of your solution, you can add folders and subfolders to your solution.

- **Clean**
  You can use the clean function to delete any runtime objects that have been created for an item or the solution. The system creates theses runtime objects when you activate an item or the solution. When you clean an item, the associated test data such as business object instances are deleted.
- **Delete**
  You can use the delete function to delete an item and its corresponding runtime objects.
- **Translation**
  You can translate language-dependent item texts such as on-screen texts and business configuration content into one of the languages supported by the SAP cloud solution. Access the translation submenu by right-clicking the project node or one of the items and selecting *Translation* *and then one of the following options*:
  - *Export Text*
    Export translatable text strings in XML Localization Interchange File Format (XLIFF).
  - *Import Text*
    Import a translated XLIFF file.
  - *Check Status*
    Check the completeness of the translation.

  For more information, see Translation Quick Guide [page 651].

Bug Fixing in Pre-production and Production Tenants

You can make changes to .absl files directly in the pre-production and production tenant to fix any bugs.

In general, if you want to make any code changes to a productive add-on, the recommended way is to create a patch in the test/development tenant where the add-on was originally developed. The patch is then uploaded to the production system only after thorough testing in the source tenant. With this feature, you can make the code changes directly in the production and pre-production system.

For more information, see Bug Fix in Production and Pre-production Tenants [page 184].

Related Information

Working in the Integrated Development Environment [page 42]

5.5.6 Add New Item Dialog Box

Overview

The *Add New Item* dialog box displays the installed item templates that SAP provides for solution development. You can add items such as business objects, different user interface (UI) components, and Web services to your solution.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio ("studio") as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Adding New Project Items

To add an item to your solution:

- In the *Solution Explorer*, right-click the project and select *Add* > *New Item*.

  > → Tip
  >
  > You can also open the *Add New Item* dialog box by clicking the button at the top of the *Solution Explorer* tool window.

  The *Add New Item* dialog box opens. By default, all items are displayed in the center pane in alphabetical order.
  You can also display items by category by selecting one of the categories from the list in the *Installed Templates* pane under *SAP*:
  - *Analytics*
  - *Business Configuration*
  - *Data Protection and Privacy*
  - *Extension*
  - *Mashups and Web Services*
  - *Screens*
  - *Service Integration*

  When you select an item, a brief description of the item appears in the right pane. You can add the following items.

Installed Item Templates — General Items

Installed Template	Icon	File Extension	Description	How To...
Business Object		`.bo`	You can create a business object definition that specifies the structure and organization of the business object. On the basis of your business object definition, you can then implement the business logic and generate screens.	... Create a Business Object [page 398]
Code List Data Type		`.codelist`	Code list data types can, for example, be used in business objects or business object extensions. You can define configurable code list data types via business configuration objects and business configuration sets.	...Create a Code List Data Type [page 472]
Form		`.xdp`	You can create a print form for a business object. Print forms are read-only portable document format (PDF) documents that are generated from data stored in the business object. You can define the data that you want to include in the form and design the layout.	... Create a Print Form [page 508]
Mass Data Run		`.run`	A run that specifies the automatic mass processing of a business object action. It contains the selection criteria and the processing results.	...Create a Mass Data Run [page 542]
Query		`.qry`	You can create a query that contains search parameters and fields from different business object nodes. You can then create an object work list (OWL) that uses your query data in search and information retrieval operations.	... Create a Query [page 421]
Reuse Library		`.library`	You can create a reuse library and define its functions to organize the code in your solution or to reuse code in several script files.	...Reuse Library Quick Guide [page 468]
Tax Decision Tree		`.bcctax`	You can create a tax decision tree for a country. Based on this decision tree, the system determines the tax event for this country when you create an invoicing document, such as a customer or supplier invoice.	...Create a Tax Decision Tree [page 394]

Installed Item Templates — Analytics

Installed Template	Icon	File Extension	Description	How To...
Analytical Report		`.report`	You can create a report based on an SAP data source or based on a custom data source you have created.	... Create a Report [page 534]

Installed Template	Icon	File Extension	Description	How To...
Combined Data Source		`.cds`	A combined data source combines data from two or more data sources that share common or similar characteristics.	... Create a Combined Data Source [page 530]
Data Source		`.ds`	You can create a data source that includes key figures and characteristics and transformation and aggregation rules. You can also reference other data sources. You can then create a report that uses your data source. To join or combine data from different data sources, you can create joined and combined data sources using the administrator tool for Analytics.	... Create a Data Source [page 527]
Joined Data Source		`.jds`	A joined data source joins data from two or more data sources that share common characteristics.	... Create a Joined Data Source [page 520]
Key Figure		`.kf`	You can create a calculated or restricted key figure based on an SAP data source or based on a custom data source you have created. You can use the key figure in a report. A calculated key figure is determined using calculation rules or formulas. A restricted key figure is restricted to a specified characteristic value.	...Create a Key Figure [page 535]

Installed Item Templates — Business Configuration

Installed Template	Icon	File Extension	Description	How To...
BAC Elements		`.bac`	You can create business options, business topics, and country options.	... Create a Business Option [page 375] ... Create a Business Topic and Business Options [page 377] ... Create a Country and Business Options [page 380]
Business Configuration Object		`.bco`	You can create a business configuration object (BCO) as the basis of a business configuration set (BC set).	... Create a Business Configuration Object [page 386]

Installed Template	Icon	File Extension	Description	How To...
Business Configuration Set		`.bcc`	You can create a business configuration set (BC set) based on a business configuration object (BCO). You can then assign your BC set to a business option using the Business Configuration Wizard.	... Create a BC Set Using an SAP BCO [page 383] ... Create a BC Set Using a Custom BCO [page 387]
Business Configuration View		`QA.uicomponent`	You can create a business configuration view (BC view) for a business configuration set (BC set) to allow administrators to change the values defined in the BC set during fine-tuning. You model the view using the UI designer.	... Create a Business Configuration View [page 389]
Implementation Project Template		`.bct`	Fine-tuning project templates provide sample content for a specific scenario that is defined by SAP. The fine-tuning activities you need to complete for the template are explicitly listed.	... Create a Fine-Tuning Project Template [page 392]

Installed Item Templates — Data Protection and Privacy

Installed Template	Icon	File Extension	Description	How To...
Personal Data Disclosure		`.pdd`	You can enable disclosure of personal data stored in custom BOs. The data protection officer can access the data in the Data Protection and Privacy work center.	... Enable Data Disclosure [page 476]
Sensitive Personal Data		`.psd`	You can define data protection and privacy level of personal data. The fields defined as sensitive personal data will be enabled for read access logging. The fields defined as personal data will be enabled for destruction during the data removal process.	... Enable Read Access Logging [page 483] ... Enable Personal Data for Extension Fields [page 485]

Installed Item Templates — Extension

Installed Template	Icon	File Extension	Description	How To...
Business Object Extension		`.xbo`	You can create business object extensions to enhance SAP business objects by adding extension fields to them. You can then add the extension fields to forms, screens, reports, and search categories. You can also use the extension fields in script files.	... Create a Business Object Extension [page 442]

Installed Template	Icon	File Extension	Description	How To...
Embedded Component		`EC.uicomponent`	An embedded component is a reusable user interface component that you can add to other floorplans. The parent floorplan influences the behavior of the embedded component. You can model this component in the UI designer.	... Model an Embedded Component [page 716]
Enhancement Implementation		`.enht`	You can create an enhancement of an SAP enhancement option to allow customers to apply business rules that meet specific requirements, for example, country-specific requirements, without modifying the standard SAP's cloud solution. An enhancement implementation consists of a message definition, a filter, and a script file.	...Create an Enhancement Implementation [page 466]
Process Extension Scenario		`.xs`	You can create a process extension scenario to link business contexts to other related business contexts. You can then use the `Scenario` annotation to specify that an extension field is available for all the business contexts in the scenario.	... Create a Process Extension Scenario [page 461]
References to Customer-Specific Fields		`.ref`	You can select the customer-specific fields that you want to use in script files.	... Create References to Customer-Specific Fields [page 453]

Installed Item Templates — Mashups and Web Services

Installed Template	Icon	File Extension	Description	How To...
Data Mashup		`.MC.uimashup`	You can create a mashup that combines and displays data from both internal and external sources. You can then use the Data Mashup Builder in SAP's cloud solution to transform or merge external Web services with internal business data using industry-standard Web service protocols.	... Create a Data Mashup [page 556]
HTML Mashup		`.MC.uimashup`	You can create a mashup that embeds HTML or JavaScript code in a screen.	... Create an HTML Mashup [page 558]
Mashup Port Binding		`PB.uimashup`	Create a port binding to add a mashup to your floorplan.	... Create a Port Binding for a Mashup [page 554]

Installed Template	Icon	File Extension	Description	How To...
REST Mashup Web Service		`.WS.uimashup`	You can create a Web service based on the REST architectural style. For REST Web services, you need to know the request URL and any relevant query parameters. You define the REST Web service using the administrator tools for mashup and Web service authoring. To integrate the Web service in a screen, you must add the Web service to a data mashup.	... Create a Mashup Web Service [page 559]
RSS/Atom Mashup Web Service		`.WS.uimashup`	You can create a Web service based on RSS or Atom feeds. For RSS or Atom feeds, you need to know the end-point URL of the Web service or the WDSL location. You define the RSS/Atom Web service using the administrator tools for mashup authoring and Web service authoring. To integrate the Web service in a screen, you must add the Web service to a data mashup.	... Create a Mashup Web Service [page 559]
SOAP Mashup Web Service		`.WS.uimashup`	You can create a Web service based on SOAP. For SOAP Web services, you need to know the WSDL location (URL). You define the SOAP Web service using the administrator tools for mashup authoring and Web service authoring. To integrate the Web service in a screen, you must add the Web service to a data mashup.	... Create a Mashup Web Service [page 559]
URL Mashup		`.MC.uimashup`	You can create a mashup that sends data from SAP's cloud solution to the URL of an online service provider. The service provider uses the data, for example, to perform a search, and the results are displayed in a new browser window.	... Create a URL Mashup [page 555]

Installed Item Templates — Screens

Installed Template	Icon	File Extension	Description	How To...
Fact Sheet (FS)		`FS.uicomponent`	An FS is a one-page read-only overview of an object. It displays the object properties, can have different views, and can contain links for navigating to related objects. You can model this overview in the UI designer.	... Model a Fact Sheet [page 697]

Installed Template	Icon	File Extension	Description	How To...
Guided Activity Floorplan (GAF)		`GAF.uicomponent`	A GAF is a floorplan for an activity that can be divided into a logical sequence of steps. It consists of a series of screens that guide the user through an activity to complete a specific task. You can model this floorplan in the UI designer.	... Model a Guided Activity Floorplan [page 708]
Modal Dialog (MD)		`MD.uicomponent`	An MD is a small movable and resizable window that helps users perform a simple task. You can model this dialog in the UI designer.	... Model a Modal Dialog [page 714]
Object Instance Floorplan (OIF)		`OIF.uicomponent`	An OIF allows users to create, delete, view, and edit a business object such as an employee, a purchase order, or a supplier's record. You can model this floorplan in the UI designer.	... Model an Object Instance Floorplan [page 702]
Object Value Selector (OVS)		`OVS.uicomponent`	An OVS supports users in selecting values for an input field. An OVS is most commonly called from an input field. You can configure an OVS component in the UI designer. You can also configure an OVS Control (an input field with a value selection helper) in the UI designer and consume the OVS component.	... Create an Object Value Selector Dialog Box [page 705]
Object Work List (OWL)		`OWL.uicomponent`	An OWL is a list pane that contains business objects such as purchase orders or opportunities. Users can perform tasks on objects and trigger follow-on processes. You can embed an OWL in a view. You can model this list in the UI designer.	... Model an Object Work List [page 703]
Port Type Package (PTP)		`PTP.uicomponent`	A PTP hosts one or more port types. A port type is a blueprint for an inport or an outport. You can create a PTP to facilitate navigation between components that have inports and outports within the same PTP without any mapping. You can configure these components in the UI designer.	... Create a Port Type Package [page 713]
Quick Activity Floorplan (QAF)		`QA.QA.uicomponent`	A QAF allows users to quickly perform a self-contained or short subtask of a larger task, for example, create a bidder within the creation of a request for quotation. A QAF is a simple task-specific alternative to an OIF. You can model this floorplan in the UI designer.	... Model a Quick Activity Floorplan [page 700]

Installed Template	Icon	File Extension	Description	How To...
Quick Create Floorplan (QC, not available for SAP Business ByDesign)		`QC.uicomponent`	A QC allows casual users to quickly create, delete, view, and edit a business object record, such as an account or an opportunity. A QC is a slide-in window, that allows users to edit the record without navigating away from the UI they are currently seeing. UIs that are created with QCs are intended to be used on tablets and on PCs.	... Generate Screens for a Business Object [page 429]
Quick View Floorplan (QV, not available for SAP Business ByDesign)		`QV.uicomponent`	A QV is a preview on a thing. A thing is the representation of one or several entities in the system and is intended to match the data record currently displayed on the UI. The QV combines several pieces of detailed information on a thing without navigating away from the UI that the user currently sees. Data displayed in a QV cannot be edited. QVs are intended to be used on tablets and PCs.	... Generate Screens for a Business Object [page 429]
Thing Inspector Floorplan (TI, not available for SAP Business ByDesign)		`TI.uicomponent`	A TI allows the user to get a complete view of a thing. The TI shows all aspects of a thing and is a compact summary of all primary attributes, the tags, and the actions that can be carried out. TIs are intended to be used on tablets and PCs.	... Generate Screens for a Business Object [page 429]
Thing Type Floorplan (not available for SAP Business ByDesign)		`TT.uidataobject`	A TT describes the general UI properties of a thing, such as its name and various icons on the UI, its binding against a business object, and its key. It is used for navigating between the screens that are thing-based and for the provisioning of REST services for a thing.	... Generate Screens for a Business Object [page 429]
UI Switch		`.uiswitch`	A UI switch is used to control role-specific changes of SAP UIs. In the UI designer, you can use the switch to restrict the visibility of SAP UI changes. Only those user roles that have the UI switch assigned in the SAP solution can see the respective UI changes.	... UI Switches [page 760]
Work Center (WoC)		`.uiwoc`	A WoC groups task-specific activities such as managing purchase orders or planning and budgeting. Each WoC has multiple views. You can model a WoC in the UI designer.	... Model a Work Center [page 712]

Installed Template	Icon	File Extension	Description	How To...
Work Center View		`.uiwocview`	A view contains information or tasks related to a specific topic. Views can contain a worklist and can link to other activity floorplans such as QAFs or GAFs. Views must be assigned to a WoC. You can model views in the UI designer.	... Model a Work Center View [page 710]

Installed Item Templates — Service Integration

Installed Template	Icon	File Extension	Description	How To...
Communication Scenario		`.csd`	You can select a set of inbound and outbound services that allows customers to create communication arrangements in SAP's cloud solution to exchange business documents electronically.	... Define a Communication Scenario [page 619]
External Web Service Integration		`.wsid`	You can create an external Web service integration of type SOAP or REST. You can then address the Web service in your business logic to enable access from the SAP cloud solution to a remote application.	... Integrate an External Web Service Using SOAP [page 612] ... Integrate an External Web Service Using REST [page 614]
Internal Communication		`.pid`	You can create a service integration definition to exchange data between two business objects that are located in different deployment units. The definition contains the configuration for asynchronous communication between the business objects.	... Configure Internal Communication [page 579]
Web Service		`.webservice`	You can create a Web service based on a business object. Web services enable remote access to the business object data using SOAP-based communication.	... Create a Web Service [page 606]
XML File Input		`.pid`	You can create an XML schema definition that allows customers to create an XML file, which can then be used to import third-party data from other systems into SAP's cloud solution.	... Create XML File Input [page 581]

5.5.7 Properties Window

Overview

The *Properties* window displays information about the item in your solution that is currently selected in the *Solution Explorer*. When you select the solution, you can view solution properties such as the solution status. When you select an item such as a business object or a UI component, you can view item properties such as the activation status of runtime objects.

Features

You can access the *Properties* window in the *View* menu by clicking *Properties Window* or by pressing **F4**. You can view the following solution properties or item properties by selecting the solution project node or an item in the *Solution Explorer*.

Solution Properties

Property	Description
Advanced	
Project File	Name of the project file: `[solution name].myproject`
Project Folder	Path of the project file in the repository
General	
Compiler Version	Version of the scripting language compiler that is used to check or activate the solution. This field typically contains two numbers separated by a slash, for example, `1302 / 1211`. The first number indicates the compiler version of the `.absl` files. The number after the slash indicates the compiler version of the `.bo` and `.xbo` files.
Created By	User who created the project
Created On	Date and time when the project was created
Deployment Unit	Name of the default deployment unit of the solution
Description	Description of the solution defined when the solution was initially created
Last Changed By	User who last changed the project
Last Changed On	Date and time when the project was last changed, that is, when an item was last added to or deleted from the solution
Name	Name of the solution generated automatically by the studio
Namespace	Container used to group all items in the solution

Property	Description
Status	Lifecycle status of the solution For customer-specific solutions, this property can have one of the following values: • `In Development` • `Assembled` • `Disabled` • `In Deployment` • `Deployed` For information about lifecycle management, see: • Lifecycle Management of Customer-Specific Solutions [page 143]
Type	Type of the solution defined when the solution was initially created; this property characterizes the focus and content of a solution. For a customer-specific solution, this property can have one of the following values: • `Customer-Specific Solution` For information, see Create a Solution [page 177]. • `Solution Template` For information, see Create a Solution Template [page 175].

Item Properties

Property	Description
Advanced	
Local Path	Full path of the project item on the client computer
Repository Path	Path of the project item in the repository
General	
Business Object Type Code (for business objects only)	Code representation of the type of the business object in the repository
Checked Out By	User who has currently checked out the project item
Check Out On	Date and time when the project item was checked out
Created By	User who created the project item
Created On	Date and time when the project item was created
File Name	The name of the file or folder

Property	Description
Last Changed By	User who last changed the project item
Last Changed On	Date and time when the project item was last changed
Status	
Activation Status	Status of the (runtime) objects of an item displayed in the *Properties* window: This property can have one of the following values: • `Runtime objects up to date` • `Runtime objects out of date` • `Runtime objects do not exist` • `Runtime objects contain errors` • `Check pending/Check not possible` Make sure that the status of the runtime objects is `Runtime objects up to date` when you perform runtime tests. Otherwise, you may test on outdated runtime data. To update the runtime objects, activate the item by right-clicking the item and selecting *Activate*. To illustrate the activation status in the *Solution Explorer*, a red dot ( ) is displayed next to the item icons whenever the runtime objects are not up-to-date. **i Note** For detailed Information about how the status of items and object items is displayed by icons in the *Solution Explorer*, see Solution Explorer [page 54].

5.5.8 Trace Explorer

The *Trace Explorer* is a tool for gathering and analyzing diagnostic information.

For more information about tracing, see Tracing Quick Guide [page 639].

Features

You can access tracing functions in the studio as follows:

- Functions in the *Debug* toolbar
  The *Debug* toolbar provides debug functions and in addition functions that allow you to start and stop tracing. You can also access the *Trace Explorer* tool window from this toolbar.
  You can access the *Debug* toolbar in the *View* menu by clicking *Toolbars* > *Debug*.
- *Trace Explorer* tool window

The *Trace Explorer* tool window displays active and completed traces. You can run a local trace, run a trace for a business user, use incident traces for troubleshooting purposes, and view the results of a recorded trace.
You can access the *Trace Explorer* tool window in the *View* menu. You can also access the *Trace Explorer* tool window from the *Debug* toolbar.

5.5.9 Working with the Wizards

Overview

The wizards in the studio guide you through a logical sequence of steps, making it easier and quicker for you to perform development activities that are complex, unfamiliar, or infrequently performed. All wizards are easy to access and follow similar user interface (UI) patterns.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio ("studio") as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Wizard Features

The wizards in the studio:

- Guide you through a logical sequence of dependent or independent working steps.
- Consist of three steps or more.
- Contain a review step.
  The *Review* step:
  - Displays all data that you entered in the wizard in the previous steps and all objects that were automatically generated.
  - Is always the last step, for example, first step (1/3), second step (2/3), *Review* step (3/3).
  - Is display-only.
    You cannot change any entries in the *Review* step. To change an entry, click *Back*.
- Remain open until you have finished defining the item and the system has generated the item.
  Most generated items appear in the *Solution Explorer*. When you define BAC elements, you can double-click the `.bac` file or right-click the file and select *Open* to display and change the BAC elements you have defined.

Each wizard step displays the following buttons: *Back*, *Next*, *Finish*, *Cancel*. The appearance of the buttons depends on the wizard step displayed, that is, whether the function is available. For example, the *Back* function is not available in the first wizard step.

Available Wizards

You can access the following wizards in the studio:

Studio Wizards

Wizard	Use
Approval Wizard	Create an Approval Process [page 495]
Business Configuration Wizard	• Create a Business Option [page 375] • Create a Business Topic and Business Options [page 377] • Create a Country and Business Options [page 380]
Business Configuration Object Wizard	Create a Business Configuration Object [page 386]
Business Configuration Set Wizard	• Create a BC Set Using an SAP BCO [page 383] • Create a BC Set Using a Custom BCO [page 387]
Code List Data Type Wizard	Create a Code List Data Type [page 472]
Communication Scenario Definition Wizard	Define a Communication Scenario [page 619]
Data Source Wizard	Create a Data Source [page 527]
External Web Service Integration Wizard	Integrate an External Web Service Using SOAP [page 612] Integrate an External Web Service Using REST [page 614]
Form Wizard	Create a Print Form [page 508]
Mass Data Run Wizard	Create a Mass Data Run [page 542]
Notification Rules Wizard	Enable Notification Rules [page 501]
Query Wizard	Create a Query [page 421]
Service Integration Wizard	• Configure Internal Communication [page 579] • Create XML File Input [page 581]
Tax Decision Tree Wizard	Create a Tax Decision Tree [page 394]
Web Service Creation Wizard	Create a Web Service [page 606]

5.5.10 Icon Library (Studio)

The following tables show some of the most frequent icons used in the studio.

Main Menu

Icon	Icon Description	Icon	Icon Description
	Save All		Save and Activate

Help Menu

Icon	Icon Description	Icon	Icon Description
	Help		Manage Help Settings

For more information, see .

My Solutions

Icon	Icon Description	Icon	Icon Description
	Log On		Log Off
	Create Solution		Refresh
	Open Solution		Delete Solution
	Customer-Specific Solution in Development		Patch Solution
	Solution Template		Multi-Customer Solution

For more information, see My Solutions [page 48].

Solution Explorer

Generic Actions

Icon	Icon Description	Icon	Icon Description
	Properties		Refresh Solution
	Add New Item		Customer-Specific Solution in Development
	Activate		Clean
	Open		Delete
	Check Out Content		Check In All Files, Check In Content

For more information, see Solution Explorer [page 54].

Solution Node Context Menu

Icon	Icon Description	Icon	Icon Description
	Add New Item		Translation Context Menu: Export Text
	Add Existing Item		Import Translation
	Add New Folder		Check Status
	For more information, see Add New Item Dialog Box [page 58].		For more information, see Translation Quick Guide [page 651].

Business Object Context Menu

Icon	Icon Description	Icon	Icon Description
	Create Script Files		Create Screens

Icon	Icon Description	Icon	Icon Description
	Node		ABSL file
	Deploy Business Configuration		Open in UI Designer

For more information, see Business Objects Quick Guide [page 398] and Add New Item Dialog Box [page 58].

Implementation Manager

Icon	Icon Description	Icon	Icon Description
	Assemble and Download Solution		Upload Solution
	Create Patch		Reset to Last Active Version
	Import Solution Template		

For more information, see Implementation Manager Quick Guide [page 162].

Trace Explorer

Icon	Icon Description	Icon	Icon Description
	Local Traces		Incident Traces
	Get Trace Details		Delete Selected Traces

For more information, see Tracing Quick Guide [page 639].

Related Information

Icon Library (UI Designer) [page 75]

5.5.11 Icon Library (UI Designer)

The following tables show some of the most frequent icons used in the user interface designer (UI designer).

For more information about what you can do with the tools, see Introduction to the User Interface Designer [page 662].

Main Menu

Icon	Icon Description	Icon	Icon Description
	Import		Save and Activate
	Display<->Edit		Activate
	Configuration Explorer		BO Browser
	Properties Window		Toolbox
	Output Window		Version History
	Extensibility Explorer		Update Metadata
	Add Flex Layout Row		Add Flex Layout Column
	Remove Flex Layout Row		Remove Flex Layout Column

Controller

Icon	Icon Description	Icon	Icon Description
	Add Operation		Remove Operation
	Bind Action		Unbind Action

Icon	Icon Description	Icon	Icon Description
	Add Parameter		Delete Parameter

Data Model

Icon	Icon Description	Icon	Icon Description	Icon	Icon Description
	Add Item		Rename		
	Structure		Dedicated Structure		Root Structure
	Data Field		Dedicated Field		Data List Field
	Data List		Dedicated Data List		

BO Browser

Icon	Icon Description	Icon	Icon Description	Icon	Icon Description
	Element		Composition Association		Intra-Business–Object Association
	Cross-Business-Object Association		Composition that allows multiple instances of the target node		Dependent-Object Association
	Intra-Business-Object Association that allows multiple instances of the target node		Cross-Business-Object Association that allows multiple instances of the target node		

5.6 Public Solution Model (PSM)

5.6.1 About the Public Solution Model (PSM)

The SAP public solution model (PSM) contains all entities in the SAP cloud solution that are released for use by external consumers. External consumers can be partners who develop solutions on top of the SAP cloud solution such as add-ons and integration scenarios or administrators who, for example, use SAP data sources to create new reports. The modular and homogenous architecture of the SAP cloud solution allows SAP to offer external consumers a highly flexible means of developing additional functionality without modifying the standard solution.

Basic Principles

Understanding PSM Entities

The *Repository Explorer* tool window displays SAP content that is released with the public solution model (PSM) such as business objects, data types, enhancement options, inbound service interfaces, and reuse UIs. You can explore the content to gain an understanding of the repository structure and the business context and use of SAP content, which you can reuse and enhance in your solution.

For more information, see Repository Explorer [page 49].

Working with the Layer Strategy

The architecture of the SAP cloud solution is structured in three layers:

- **Presentation** — the user interface
- **Application** — the business logic
- **Persistence** — the database and search index

The SAP Cloud Applications Studio (“studio”) allows SAP partners, resellers, and customers to work with the same entities that SAP uses in the core development of its cloud solutions and to develop solution capabilities that have the same look and feel as the SAP standard cloud solutions. The content developed using the studio resides on a different layer to the SAP content; therefore, SAP content that is released with the public solution model (PSM) can be reused or enhanced without modifying the standard cloud solution. This content can be released for read-only access or for read/write access.

Enhancing SAP Content

External consumers can extend or include additional business logic in a variety of SAP entities to enhance the business functionality of the solution. Examples are new extension fields in business objects, which can be

included in user interfaces, reports, extension scenarios, or new embedded components that can be included in standard SAP user interfaces to enhance the screens.

Reusing SAP Content

External consumers can reuse specific SAP content without modifying the standard solution. For example, external consumers can reuse complete SAP user interfaces and enable navigation from a custom floorplan to an SAP floorpan and from an SAP floorplan to a custom floorplan. It is also possible to reuse SAP application logic including application logic that is relevant for legal requirements, for example, by reusing parts of SAP business objects or data types.

Understanding Deprecation

The current structures and processes defined within the public solution model (PSM) remain stable for the current and upcoming release. If it is necessary for SAP to make changes to PSM entities to meet technical or business requirements, SAP uses a deprecation mechanism that allows external consumers to update their solutions in an appropriate time frame.

Example

SAP plans to make changes to specific business object elements in the future:

- The business object elements are set to *deprecated* and, if an element is deprecated, all children are also deprecated. For example, if a business object node is deprecated, all actions, queries, and elements that belong to this node are also deprecated.
- SAP identifies the use of deprecated elements in a solution, informs the partner or other external consumer who developed the solution, and provides information about the successor element.
- The external consumer creates a new version of the solution before another SAP release becomes available.
- The new version of the solution is made available to customers for activation with the new release of the SAP solution
- SAP revokes the respective business object element.

Requesting Release of Additional PSM Entities

If you need to access business content for the SAP cloud solution that has not been released for consumption through the PSM, you can request SAP to release this business content. To do this, proceed as follows:

1. Create an incident in the SAP cloud solution and use the prefix **PSM**.
   For information on how to create an incident, see here [page 628].
2. Describe the use case of your solution and what you are trying to achieve (that is, what is the purpose of your development/solution?).
3. Describe the required business content/entities to be released in PSM, and why you need them.

4. If you require business content such as the fields of a business object from a UI, provide a screenshot of that UI and highlight the elements that need to be enabled for read-access and for write-access.
5. Add your contact information - that is, e-mail address and telephone number - so that SAP can easily get back to you in case of questions.

SAP Content Types

Deployment Units

A deployment unit is a piece of software that can be operated on a separate physical system, isolated from other pieces of software. the SAP cloud solutions are organized in deployment units that group semantically related business objects. For example, the Customer Relationship Management deployment unit groups business objects such as Campaign, Lead, Opportunity, and Sales Order. The Foundation deployment unit mainly contains master data objects, such as Material, Employee, and Customer.

Business Objects

A business object is a code representation of a self-contained, independent, real-world business concept, such as a company, an employee, a sales order, an opportunity, or a product. The definition of a business object describes the structure and behavior of a real-world business concept. Each business object resides in a deployment unit and consists of a hierarchy of nodes, which represent the data.

Business object elements can be published for read-only access or for write access; write access means that an external consumer can create, update, or delete instances of that business object by calling the core services `Create`, `Update`, or `Delete`.

Data Types

A data type is a classification of a particular type of information.

Enhancement Options

An enhancement option is a repository object provided by SAP for a specific application to allow the standard behavior of the application to be enhanced without modifying the standard solution. Enhancement options are defined with metadata and documentation. A distinction is made between single-use enhancement options and multiple-use enhancement options. If an enhancement option is for single use, it can be implemented only once in a solution. If an enhancement option is for multiple use, it can be implemented more than once in a solution.

For information about creating an implementation of an SAP enhancement option, see Create an Enhancement Implementation [page 466].

Inbound Service Interfaces

An inbound service interface is an interface used to query or manage certain entities or a group of entities within the SAP solution from an external application. Examples of inbound service interfaces are B2B messages or web services. Inbound service interfaces consist of inbound service interface operations which offer specific business functionality for an external consumer.

Embedded Components

An embedded component is a reusable user interface component that you can add to other floorplans. The parent floorplan influences the behavior of the embedded component.

Object Value Selectors

The object value selector (OVS) is a user interface (UI) pattern that supports users in selecting values for an input field. It helps users to select one item from a table of business object instances. It is most commonly called from an input field by means of a selection dialog icon. If an input field contains data, it is transferred to the selection modal dialog. Unlike other input and value help controls, an OVS does not appear attached to an input field; it appears within a modal dialog that floats above the entire window. It is possible to save and organize queries. Examples of SAP object value selectors are Employee, Campaign, and Sales Order.

Business Adaptation Catalog (BAC) Elements

The full set of the SAP cloud solution's capabilities are outlined in a central business adaptation catalog (BAC). This catalog organizes and structures the capabilities into a hierarchy of business areas, packages, topics, and options. Solutions created in the studio require business configuration content that then appears as elements (BAC elements) in the catalog and allows administrators to implement solutions in the production environment.

For more information, see Business Adaptation Catalog [page 370].

Data Sources

A data source is an object that provides a multidimensional, analytical view of business data. Data sources are the basis of analytical report content and are associated with a specific access context or can be unrestricted.

Reuse Libraries

- Basic reuse libraries
  The basic reuse libraries of the scripting language extend the scope and functions that you use to implement your business logic in the studio. These “language-near” libraries provide basic functions that are used quite often, for example, to retrieve context data, such as the current date or time for the current identity.
  For more information, see Basic Reuse Libraries Reference [page 274].
- Reuse libraries for business areas
  The reuse libraries for business areas provide additional, very specific services to the scripting language that are relevant for functions in the SAP solution's business areas, for example, in the Financial Management business area. These services are all based on entities provided by SAP.
  For more information, see Reuse Libraries for Business Areas Reference [page 308].

Extension Scenarios

Extension scenarios enable you to create an extension field in one business context and then add that extension field to multiple other business contexts. The data stored in the extension field is then passed from one business context to the next as part of a business process. The following extension scenarios have been predefined by SAP:

- Account - General Information to Lead - General Information
- Account - General Information to Opportunity - General Information
- Lead - General Information to Opportunity - General Information

5.7 Introduction to Business Object Modelling

5.7.1 Business Object Modeling

Overview

The SAP cloud solutions are built using the concept of “business objects” to model the business environment. This document provides an introduction to the structure and behavior of SAP business objects.

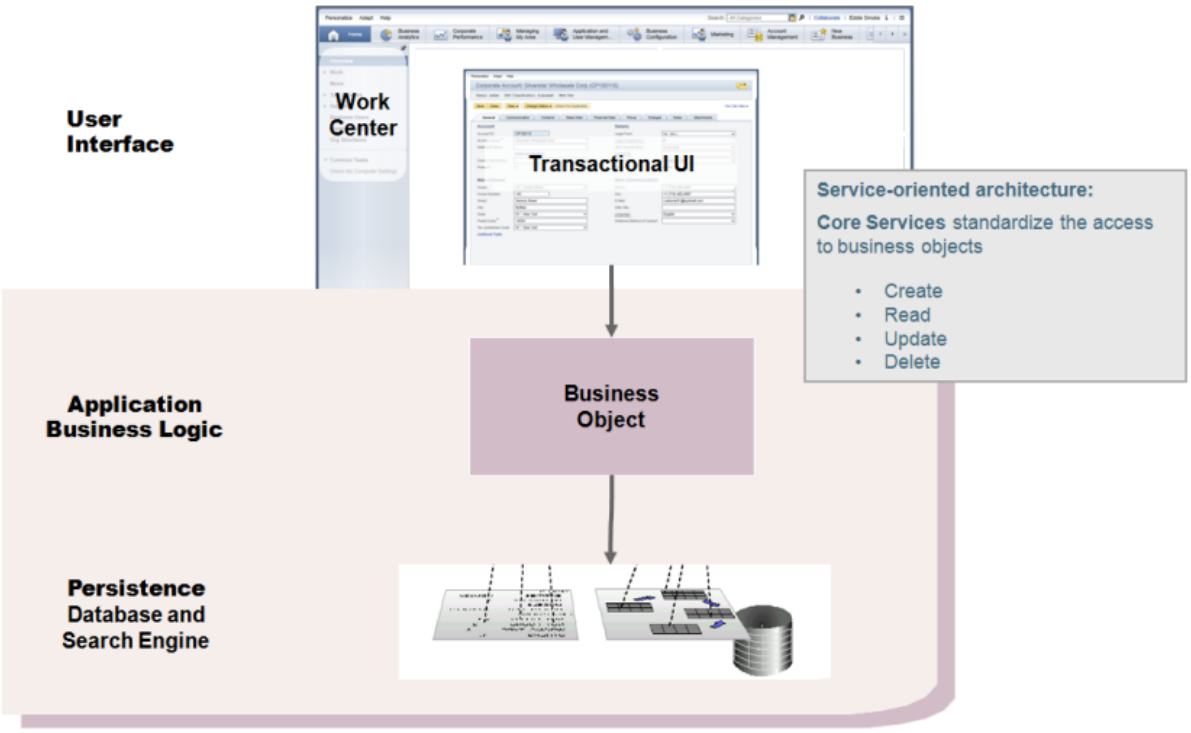
Architecture of the SAP Cloud Solution

The cloud solutions are based on a 3-tier architecture:

- User Interface
- Business Logic

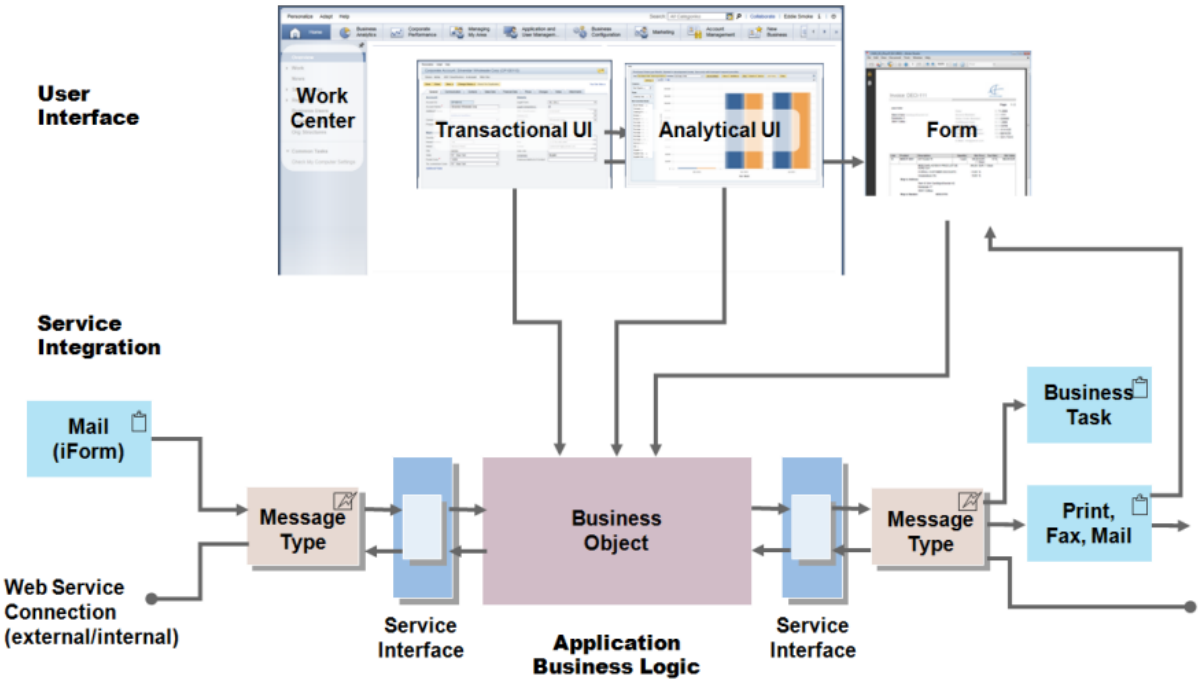
- Persistence Layer (database, search index)

All business logic is implemented in business objects that model real-world objects and processes.



The business object model is the central anchor for script coding, the user interface (UI), forms, and business tasks.

> → Tip
>
> Changes to the business object may require corrections to scripts, UI screens, forms, and so on. Therefore, when developing an cloud solution, we recommend that you complete the business object design as far as possible before proceeding.

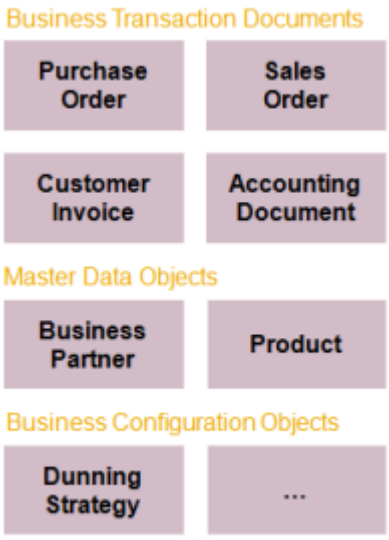


What is a Business Object ?

A business object is a self-contained, independent business concept that is well known in the business world. Typical business objects are Sales Order, Employee, and Product.

In the cloud solutions, business objects:

- Encapsulate business logic and business data
- Act as a service consumer and provider
- Are accessible exclusively through a standardized set of (core) services
- Are described by a business object model that defines:
  - Structure
  - Type of the attributes
  - Aspects of the behavior
  - Service interfaces



> **Note**
>
> The business object is the type and not the instance:
>
> - business object = purchase order
> - business object instance = purchase order with ID 32452

Business Object Categories

Business objects are classified into categories. For more information, see here [page 97].

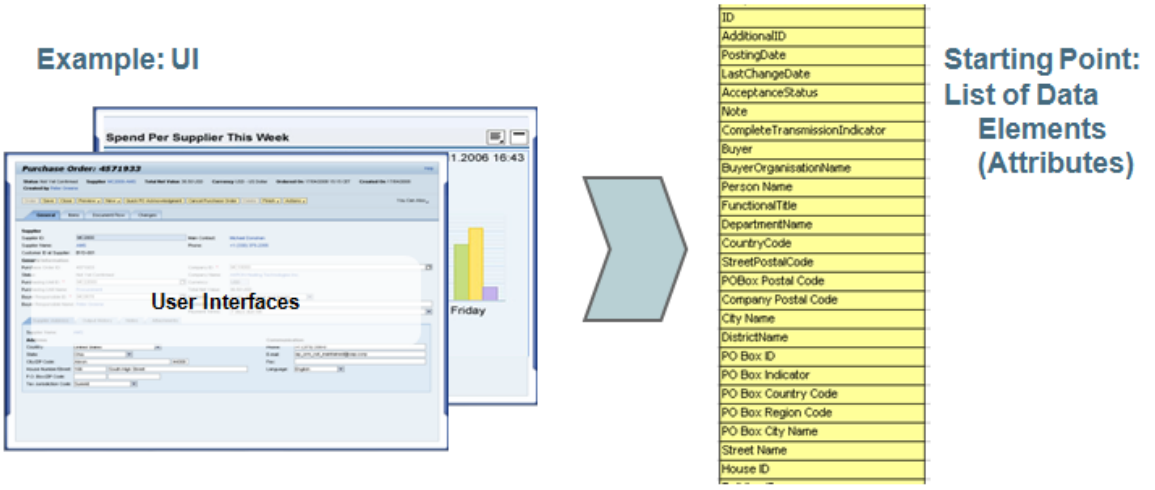
Deployment Unit

Each business object belongs to a deployment unit, which is a piece of software that can be operated on a separate physical system, isolated from other pieces of software. For more information, see here [page 205].

Creating your Business Object

There are three starting points for the development of a business object model:

- The user interface (UI), which represents all the information the user has to interact with (such as data entry, analytics, and reporting).
- An integration scenario with the message choreography, for example with SAP or a business-to-business partner. Each message represents a business document with more or less structured information.
- The engines that provide services concerning a specific topic (such as a pricing engine or tax engine).



Determining and Naming the Objects

By analyzing the business scenario you want to model, you determine which processes and objects should be modeled as separate business objects. Information that describes the business object in more detail is modeled as elements within the business object.

To ensure the right naming according to the business semantics, the business object name is constructed / chosen according to ISO 11179 naming standards.

- The names are in British English
- Abbreviations and acronyms are allowed only when used normally within business terms

Creating the Structure

The internal structure of a business object (BO) is described by nodes, which are connected by relationships. The creation of a node is primarily based on the business understanding of the BO.

For a simple BO, you may decide not to create any nodes, in which case all elements belong to the root node of the BO, which is created implicitly.

A separate node represents a logical / semantic group of elements that can be treated as a group even if the elements could be assigned directly to the higher-level node.

- Elements in one node can be mandatory or optional.
- Elements with multiplicity larger than 1 require a separate node.
- Nodes are related by a strict hierarchical relationship
  - Compositions with multiplicity
  - No n:m relationship

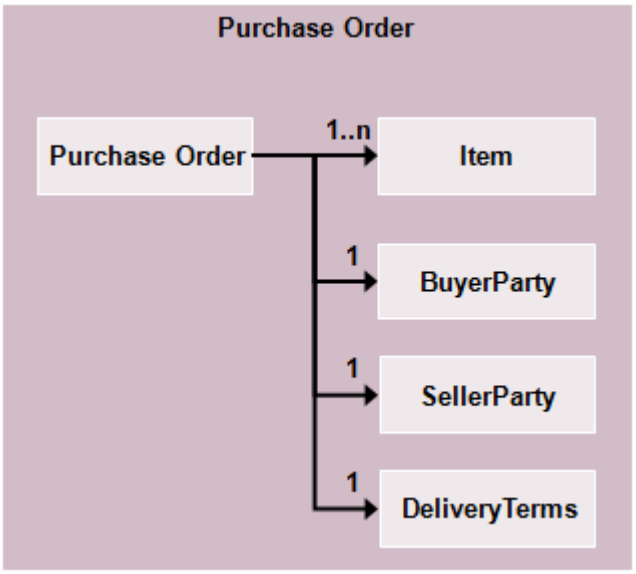
The creation of a node implicitly introduces a composition, that is, a relationship between the parent node and the child node. Compositions can be used to access features of the child node from the parent node (for example, in action implementations). For each composition, a reverse association to the parent node is created implicitly.

Example: Purchase Order

A Purchase Order is a buyer's request to a seller to provide or deliver certain quantities of products at one or several dates.

We can model the purchase order with the following nodes:

- Item – we can have one or more items, so this node is given a multiplicity of [1,n]
- Buyer – the buyer of the goods or services
- Seller – the seller of the goods or services
- Delivery Terms



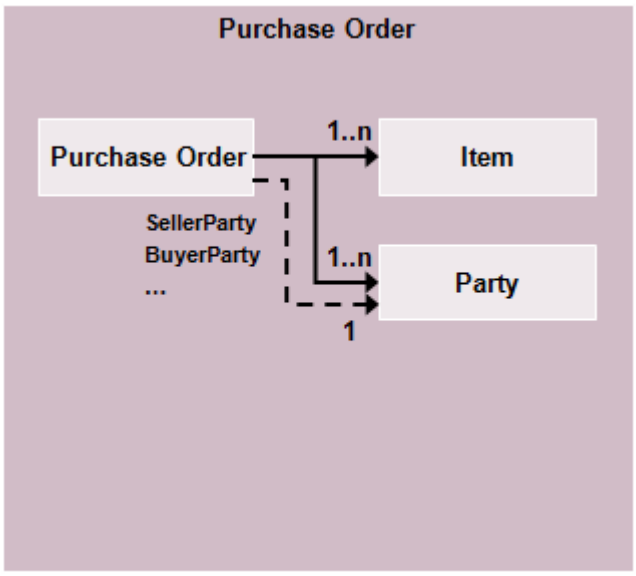
Creating Sub-types

An object or object node can "play" different roles. You have to decide which roles are important from a business point of view and should be represented explicitly by sub-types.

Analyze all objects and object nodes to determine whether some roles can be merged – this is an indication for the existence of sub-types. Merging the roles leads to generalized objects or components.

Example: Purchase Order – Sub-types

In our Purchase Order we can see that Buyer and Seller are two types of the more general component "Party". There could also be other parties involved in this transaction, such as a Bill To party.



Internal Association for Navigation

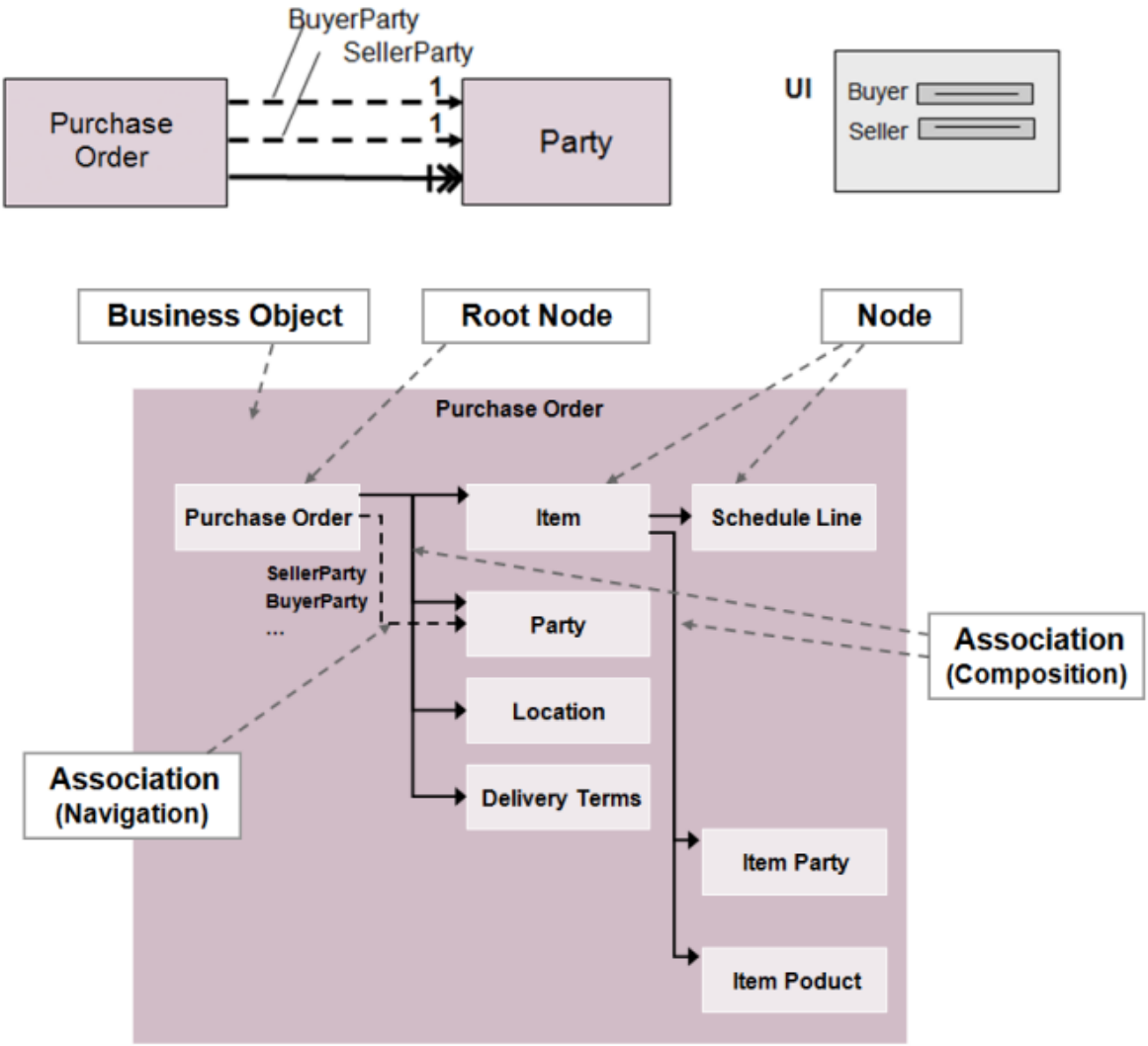
Internal association (also referred to as intra-BO association) enables navigation between nodes that is not provided directly by structural relationships.

Association for Navigation:

- Enables you to create a “navigation shortcut” to facilitate / optimize navigation
- Can be used where the target object node is determined by algorithm (“calculated / derived” relationships):
  - Navigation to multiple instances (“range”) – target multiplicity: [0,n] or [1,n]
  - Navigation to a single instance – target multiplicity: [0,1] or [1,1]

Example: Purchase Order – Internal Association

In our purchase order we can use association for navigation to simulate specialization by role.

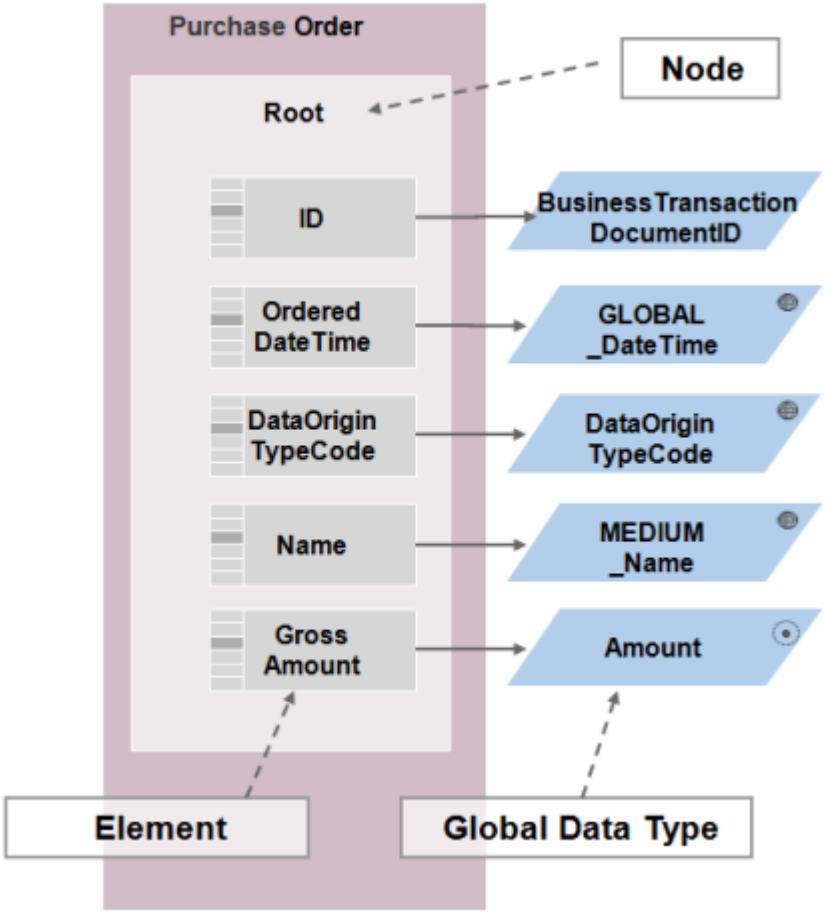


Creating the Elements

Once you have defined the business objects and nodes you can assign elements of a particular data type to the nodes.

To homogenize the use of data types, SAP defines global data types that represent business-related content in conformance with widely used Web and business standards, such as DateTime, Amount, Currency, Text, String, and Boolean.

All business objects and service interfaces share the same pool of global data types. For more information, see Data Types [page 92].



Creating External Associations

External association (also referred to cross-BO association) is used to build relationships between business objects. These relationships are classified as aggregations and associations and are described from a business point of view.

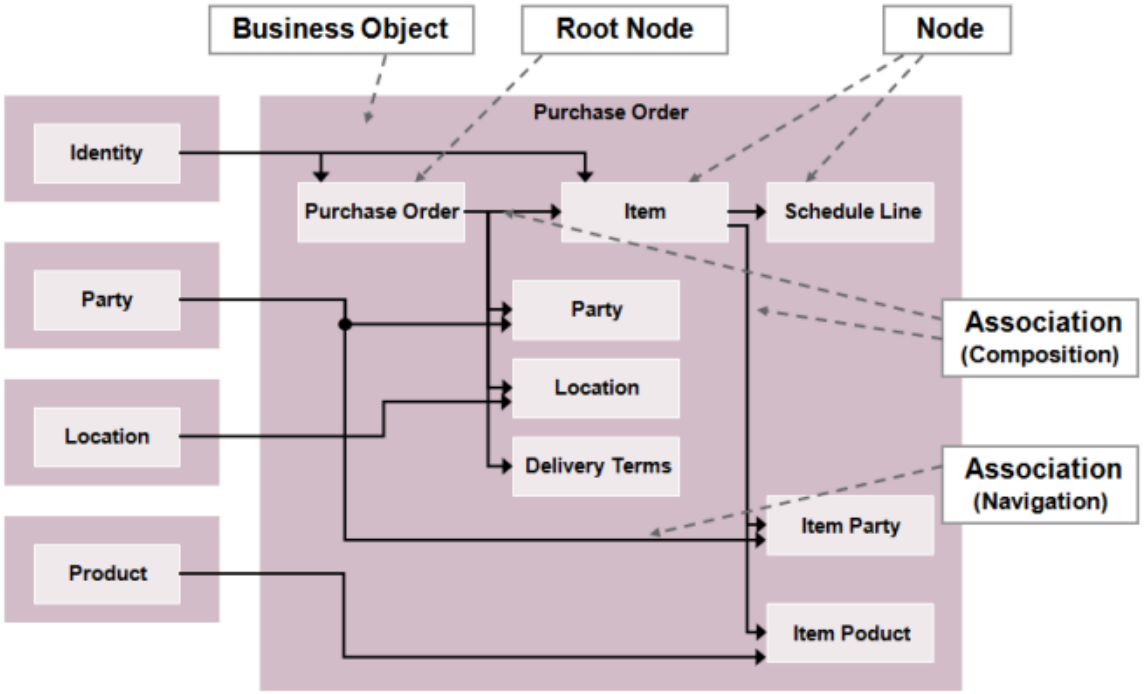
Attributes with a name related to another object are replaced by the ID of the referenced object. The attributes are then removed from the current structure and included in the referenced object. This integration produces a local view of the object model.

Example: Purchase Order

The Purchase Order business object model contains:

- Party
- Product
- Location

In the element structure these associations are represented by the corresponding reference component.



Defining the Behavior

Business object nodes offer services (operations) that can be accessed using interfaces. Operations and their implementation are generated from the business object definition.

Business logic is added in script coding, which may call operations from other business object nodes.

Core Services and Actions

Core Services

The core services of Create, Retrieve, Update, and Delete ("CRUD") are automatically handled by the studio.

Save

The Save service checks the consistency of the business object and saves to the database. This is automatically handled by the studio.

Create with Parameters

This service creates a business object or business object node instance based on specific input parameters. For example:

- CreateWithReference (for example, create sales order from sales quote)
- CreatePaymentCard (for example, for a sales order)
- Copy

Status and Action Management

Status actions check the data of the business object and transfer the business object to another status, or refuse the requested status change. For example:

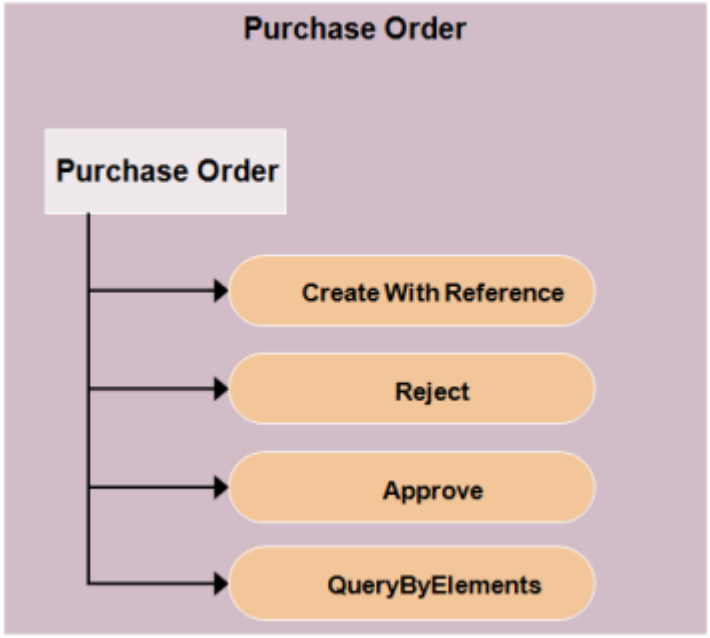
- Release
- Approve
- Complete

The status of a business object can determine which actions are allowed.

Queries

A query is a service that returns a list of business object instances based on query parameters. Queries support sorting, paging, and the use of "search engine" indexes.

- Simple queries for all elements in a business object node (QueryByElements) are automatically handled by the studio
- Complex queries can also be created that allow joining over business object nodes and business logic (for example, complex time dependencies)



Related Information

Example: Building a Solution [page 99]

5.7.2 Business Object Modeling: Data Types

Each element of a business object must be assigned a data type. To homogenize the use of data types, SAP defines consolidated global data types (GDTs) that represent business-related content in conformance with widely used Web and business standards. All business objects and service interfaces share this same pool of GDTs.

> **i** Note
>
> You can access the data types that are available via the Repository Explorer [page 49]: *Data Types*.

SAP data types are located in the **namespace**: `AP.Common.GDT (namespace: http://sap.com/xi/BASIS/Global)..`

Data Types

The studio uses the following data types:

Basic Data Type

The basic data type does not contain any elements.

Example

Enterprise Search Name	Enterprise Search Namespace	Initialization
`CurrencyCode`	`http://sap.com/xi/BASIS/Global`	`element <currency code> : CurrencyCode = "EUR";`

Structured Data Types

There are two different structured data types. The first structured data type consists of content and one or several attributes.

Example

Enterprise Search Name	Enterprise Search Namespace	Initialization
`Amount`	`http://sap.com/xi/AP/Common/GDT`	`element <amount> : Amount = {content = "15", CurrencyCode = "EUR"};`

The second structured data type consists of content and technical attributes. These are hidden in the studio so that only the content is visible.

Example

Enterprise Search Name	Enterprise Search Namespace	Initialization
`ProductID`	`http://sap.com/xi/AP/Common/GDT`	`element <identifier> : ProductID = {content = "MCP-100"};`

Aggregated Data Type

The aggregated data type consists of several basic or structured data types. It does not support labels, tooltips, or default values. A typical example of an aggregated data type is Price.

Example

Enterprise Search Name	Enterprise Search Namespace	Initialization
`Price`	`http://sap.com/xi/A1S/Global`	`element <price>: Price;`

Identifier

You use the `ID` data type to identify a business object or business concept.

```
element <ID_name> : ID;
```

The `ID` data type has up to 60 characters, is not structured and does not contain the `Content` element. The data type supports, for example, upper case conversion and alpha conversion. Alpha conversion fills purely numeric user input like “4711” with leading zeros to allow better sorting in character fields. For example, if alpha conversion is not used, an alphabetic sorting would lead to results such as: “1”, “10”, “100”, “1000”, “2”, “20”, “200”, and so on.

String

Instead of a “string” data type, you use the data types `Name`, and `Description`, which are technically strings.

You use `Name` as the short text for a business object, and `Description` as the long text.

For rapid prototyping, you can use language-independent `Name`, `Description`, and `Text` to avoid language handling, which costs extra effort in coding and on the user interface.

```
element <name> : LANGUAGEINDEPENDENT_SHORT_Name;
element <description> : LANGUAGEINDEPENDENT_LONG_Description;
```

```
element <additional_text> : LANGUAGEINDEPENDENT_EXTENDED_Text;
```

The length of the string can be one of the following:

- SHORT = 10 characters
- MEDIUM = 40 characters
- LONG = 80 characters
- EXTENDED = 255 characters

Boolean

You use the `Indicator` data type to identify a Boolean element.

```
element <some_indicator> : Indicator;
```

Date and Time

You use the following data types to identify dates and times:

```
element <some_date> : Date;
element <some_time> : Time;
```

Example

```
element <some_date> : Date = "2012-04-24";
element <some_time> : Time = "16:35:14";
```

Number

You use the following data types to declare numbers, amounts, quantities, and measures:

```
element <some_number> : NumberValue;
element <some_amount> : Amount;
element <some_quantity> : Quantity;
element <some_measure> : Measure;
```

- Use the `NumberValue` data type for integers.
- Use the `Amount` data type for money. This is a structured data type that consists of a number and a currency code (for example: 25 EUR).
- Use the `Quantity` data type for all other quantities (pieces, length, weight, and so on). This is a structured data type that consists of a number and a code for the measurement unit (for example: 25 KGM represents 25 kilograms).

- Use the `Measure` data type for measures (pieces, length, weight, and so on). This is a structured data type that consists of a number and a code for the measurement unit (for example: 25 MTR represents 25 meters).

Percent and Ratio

You use the following data types to declare percent and ratio values:

```
element <some_percent> : Percent;
element <some_ratio> : Ratio;
```

Code Lists

A code list is a field with a set of predefined values. SAP provides a large number of code lists as data types that you can use. The data type name ends with the word “Code” to identify it as a code list.

Some common code lists are:

```
element <priority_code>    : PriorityCode;
element <release_status_code> : ReleaseStatusCode;
element <approval_status_code>    : ApprovalStatusCode;
```

You should use an SAP code list where possible. If no appropriate SAP code list is available, you can define your own code list data type with your own code list in the studio. There are two ways to create code lists:

- Static code lists. For more information, see Create a Code List Data Type [page 472]
- Configurable code lists. For more information, see .

Most SAP code lists are context-independent. Some of the code lists, however, are context-dependent, that is, the code values differ depending on the context (for example, country).

The following table shows the values of the `WorkAgreementAdministrativeCategoryCode` code list of two different countries.

List ID	Country Code: United States	Country Code: Germany
1	Hourly	Hourly-Paid Industrial Worker
2	Salaried Non-Exempt	Salaried Employee
3	Salaried Exempt	Senior Executive
4		Monthly-Paid Industrial Worker

For more information, see

E-mail Address and Web Address

You use the following data types to declare e-mail and Web addresses:

```
element <e-mail_address> : EmailURI;
```

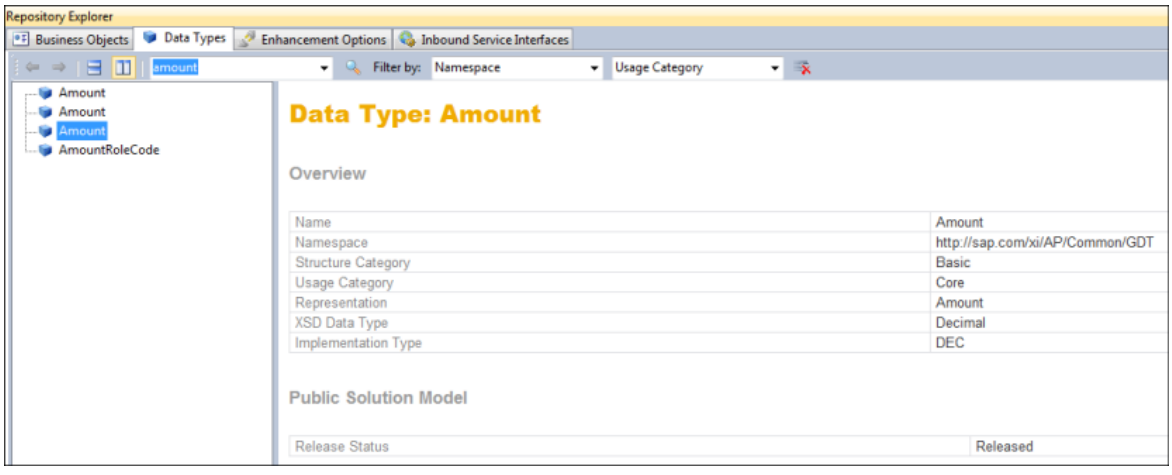
```
element <web_address> : WebURI;
```

> **Example**
>
> ```
> element  <some_email_address> : EmailURI = {content = "foobar@sap.com"};
> element <some_web_address> : WebURI = "http://www.sap.com";
> ```

5.7.3 Look Up the Representation Term of a Data Type

Procedure

1. In the studio, select *View* > *Other Windows* > *Repository Explorer*.
2. In the *Repository Explorer*, select *Data Types*.
3. In the *Search data types* window, search for the data type you want to use.
   The data type details window opens.
4. Under *Overview*, you find the representation of the data type.



Example: Looking Up the Representation Term of a Data Type

Related Information

Repository Explorer [page 49]

5.7.4 Business Object Categories

Business Transaction Document Objects and Master Data Objects

Business transaction documents and master data objects are both used to represent self-contained, independent business concepts, that are familiar to the business world (for example, in an international standard or industry best practice).

Examples of business transaction documents:

- Campaign
- Opportunity
- Service Request

Examples of master data objects:

- CostCentre
- Customer
- Employee
- SalesArrangement

Business Configuration Object

A business configuration object represents business configuration content that is not changed during normal operations. These objects may contain active content and inactive content (new implementation or business configuration change project).

Example of business configuration object:

- BusinessTransactionDocumentItemTypeCountry

Mass Data Run Object

A mass data run is the automatic and parallel processing of a typically huge number of selected objects in accordance with given business rules. A mass data run object controls mass data runs on a selected set of objects.

Examples of mass data run objects would be:

- Material and Resource Planning (MRP) Run
- Dunning Run

Technical Object

A technical object supports the technical infrastructure or IT Service and Application Management (ITSAM) of the application platform.

Example of technical objects:

- ObjectIdentifierMapping

Transformed Object

A transformed object provides consumer-friendly services across multiple business objects and their already existing functionality. A transformed object is used if a view on multiple business objects is required in the representation of a business object (for example, because the using application requires a business object). Transformed objects are instantiated at run-time and do not have their own persistence – they rely on the persistence of the underlying business objects.

Example of transformed objects:

- BusinessDocumentFlow

Dependent Object

A dependent business object is used as a reusable part in another business object. It represents a concept that cannot stand alone from a business point of view. Instances of dependent objects can only occur in the context of other business objects.

Examples of dependent objects:

- Attachment Folder
- PartnerAddress
- PriceSpecification
- Text Collection
- Workplace Address

Template Object

A template object is used to describe objects representing similar subject matter. It specifies the union of all components (nodes, relationships, elements, service operations) of the objects, without any redundancy.

Examples of template objects would be:

- Customer Transaction Document (this is the template for SalesOrder, Quote, Return, ServiceOrder, ServiceRequest, and so on)
- Product (this is the template for Material, Service Product, Warranty, and so on)

5.8 Example: Build a Simple Solution

5.8.1 Example: Building a Solution

In this example you are going to build a solution to manage the allocation of car parking spaces to employees in the SAP cloud solution.

- The solution contains one business object – representing a car park – of which there can be multiple instances.
- The car park business object contains a node for "parking spaces" – of which there can be 0 to n.
- Each parking space is assigned to an Employee ID using an Object Value Selector (OVS) – this allows you to look up the employees within the company.
- The parking space node has an association to the Employee business object – this allows you to retrieve the employee's first and last name.

> ⚠ Caution
>
> Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

Tasks

Create the Business Object

1. Open the studio and log on to the repository.
2. In the *My Solutions* tool window, right-click *My Solutions* and click *Create Solution*.
   The *Create Solution* dialog box opens.
3. Enter a *Description* for your solution, select a deployment unit and *Customer-Specific Solution*, then click *OK*.
4. In the *Solution Explorer*, right-click the project and select *Add* > *New Item*.
5. Select the *Business Object* template and enter a *Name* (for example **`CarPark`**), then click *Add*.

> i Note
>
> To add data to your business object, you declare an "element", enter a name, and assign it a data type. Elements are entered according to the following syntax:
>
> ```
> element <your_element_name> : <data_type>;
> ```
>
> When you type the colon (`:`), the code completion function presents a list of valid data types, from which you can select.

6. Add some data to the car park. First you need an ID to identify your car park. You can then add extra information, such as a name and the number of car spaces.

```
element CarParkID : ID;
element CarParkName : LANGUAGEINDEPENDENT_LONG_Text;
element CarParkDescription : LANGUAGEINDEPENDENT_LONG_Description;
element Number_of_ParkingSpaces : IntegerValue;
```

7. Each car park can have multiple car parking spaces. Therefore, you create a node with a multiplicity of [0,n] to represent the car spaces within each car park. Each node requires an ID.

```
node ParkingSpace [0,n]    {
  element ParkingSpaceID : ID;
  }
```

8. Each car space can be assigned to an employee. Therefore, you create an element of type EmployeeID to record the employee.

```
element ParkingSpace_AssignedEmployeeID : ID;
```

9. To make the connection to the existing employees in the company, you create an association to the SAP business object: Employee. This enables you to use the data from this business object in your screens. To use the Employee association you first have to import the namespace of the Employee business object.

```
import AP.FO.BusinessPartner.Global;
```

```
association ToEmployee [0,1] to Employee;
```

10. You can add some additional information for the car space.

```
element ParkingSpace_Disabled : Indicator;
element Vehicle_RegistrationNumber : LANGUAGEINDEPENDENT_MEDIUM_Text;
element ParkingSpace_Level : NumberValue;
```

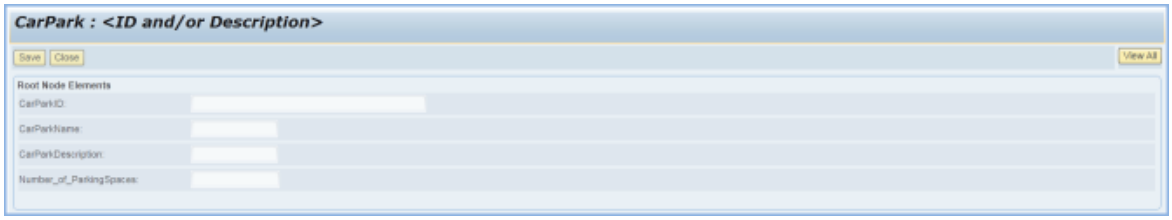
11. Save and activate the business object.

You can find the complete business object code here [page 114].

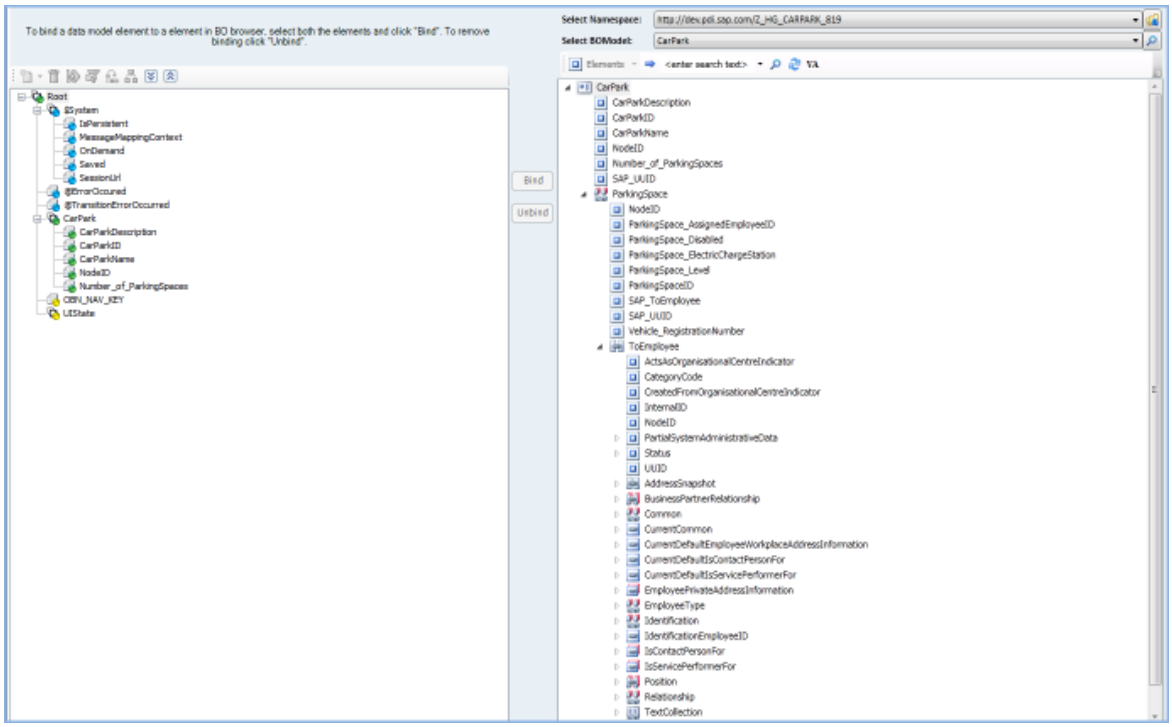
Create the Screens

The next step is to create the screens that will display the data from the business object, and allow the user to interact with the business object.

1. Right-click your business object and select *Create Screens*.
   The *Create Screen* dialog box opens.
2. Enter a *Name* for your screens, select a scenario, enter a *Short ID* and click *OK*.
   The default screens provide a starting point for your solution and include built-in navigation between screens. For this example, you will use the default Object Work List (OWL) and you will modify the Quick Activity Floorplan (QAF) screen.
3. In the *Solution Explorer*, double-click the `QA` floorplan to open it in the user interface designer (UI designer). On the *Designer* tab, you can see that the root node elements from the business object are automatically displayed as fields on the screen.



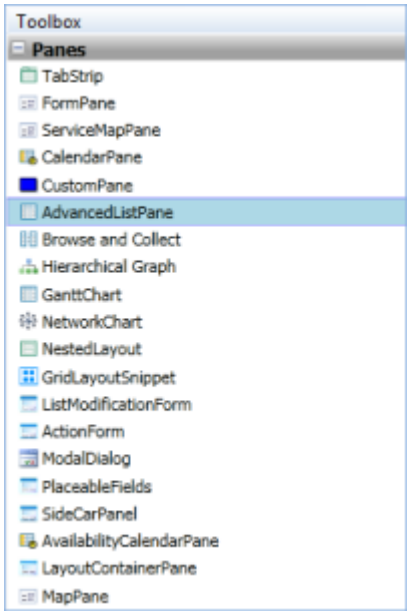
On the *DataModel* tab, you can see the data fields to the left, and on the right you can see the structure of your business object, including the `ParkingSpace` node and the association to the SAP business object `Employee`.
Notice that data fields are automatically created for the root node elements. Selecting one of the data fields shows that a binding exists with the corresponding business object element.



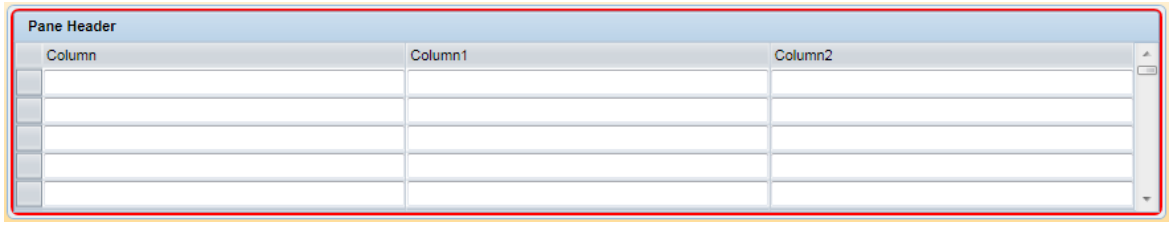
Create the Data List

For each car park, you want a table listing the car spaces and the assigned employee for each space.

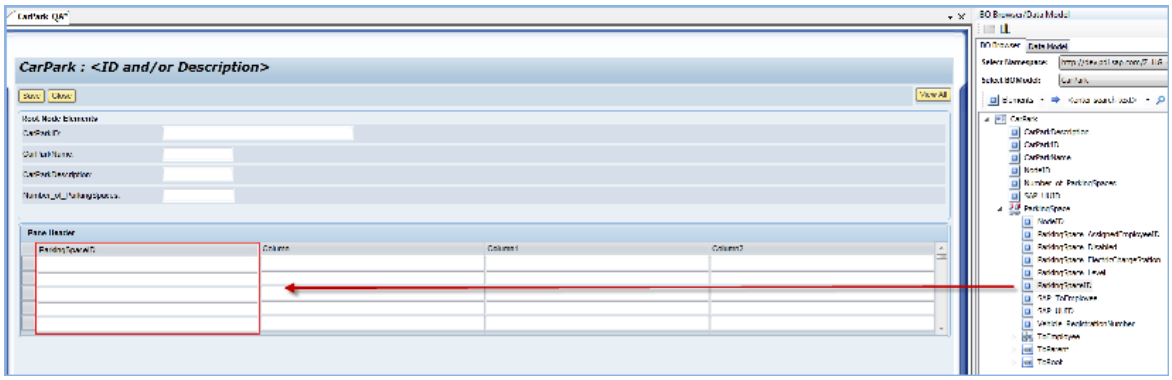
1. Open the *Designer* tab, click the `AddFlexLayoutRow` button and select the new row.
2. On the *Toolbox* tab, under *Panes*, select `AdvancedListPane` and drag and drop it onto the new row.



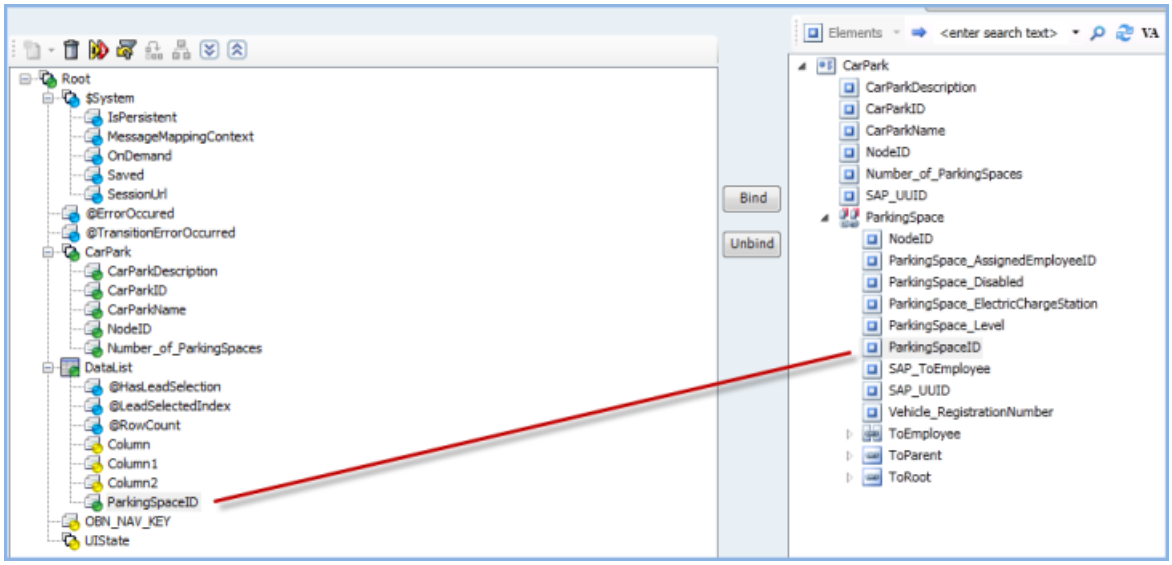
The list pane is displayed on the *Designer* tab.



3. Right-click one of the columns and select *Add* > *Column*.
4. Open the *BO Browser/Data Model* by selecting *View* > *BO Browser* or clicking the *BO Browser* button.
5. In the BO Browser, select the `CarPark` BO Model.
6. Drag the *ParkingSpaceID* element onto the first column header of the list pane. Select *No* when prompted to use backend text and save your changes.



On the *DataModel* tab, you can see that the data field and the binding are created automatically.

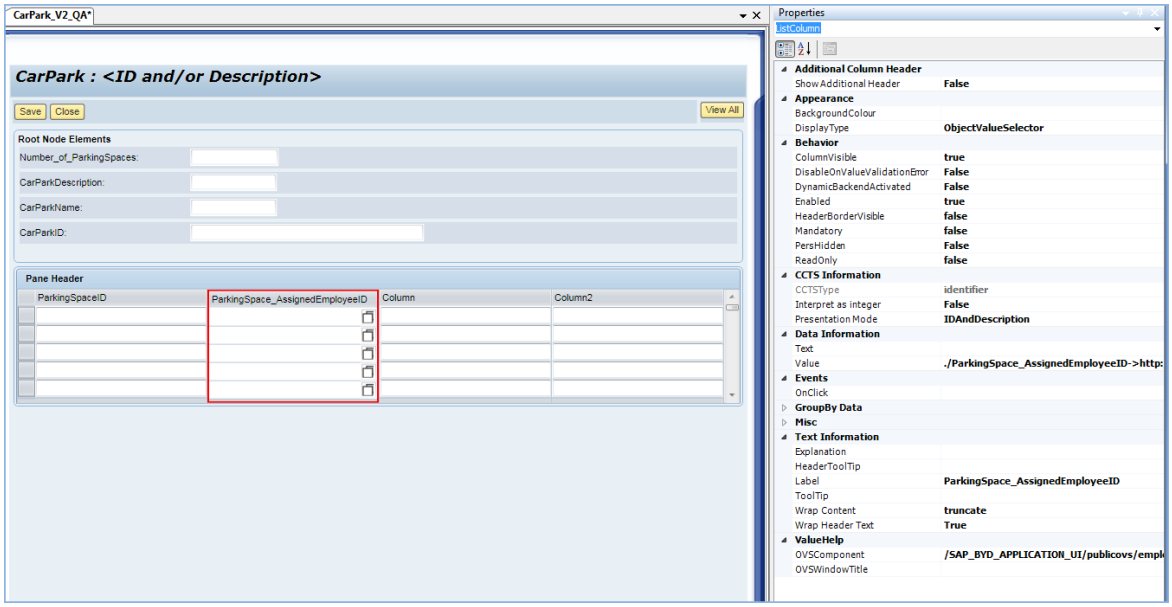


7. On the Designer, drag the *ParkingSpace_AssignedEmployeeID* element onto the second column header of the list pane. Select *No* when prompted to use backend text and save your changes.

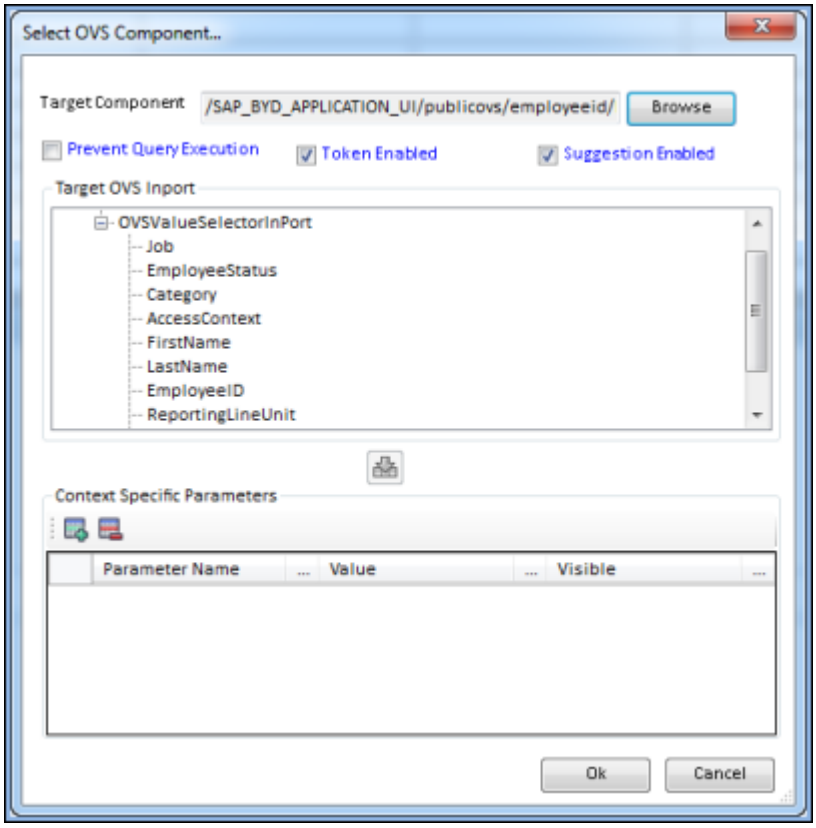
Add the Object Value Selector

You can use an Object Value Selector (OVS) to select the employee ID from a list of the existing employees.

1. Select the *ParkingSpace_AssignedEmployeeID* column and in the *Property Explorer*, under *Appearance*, select *ObjectValueSelector* as the *DisplayType*.



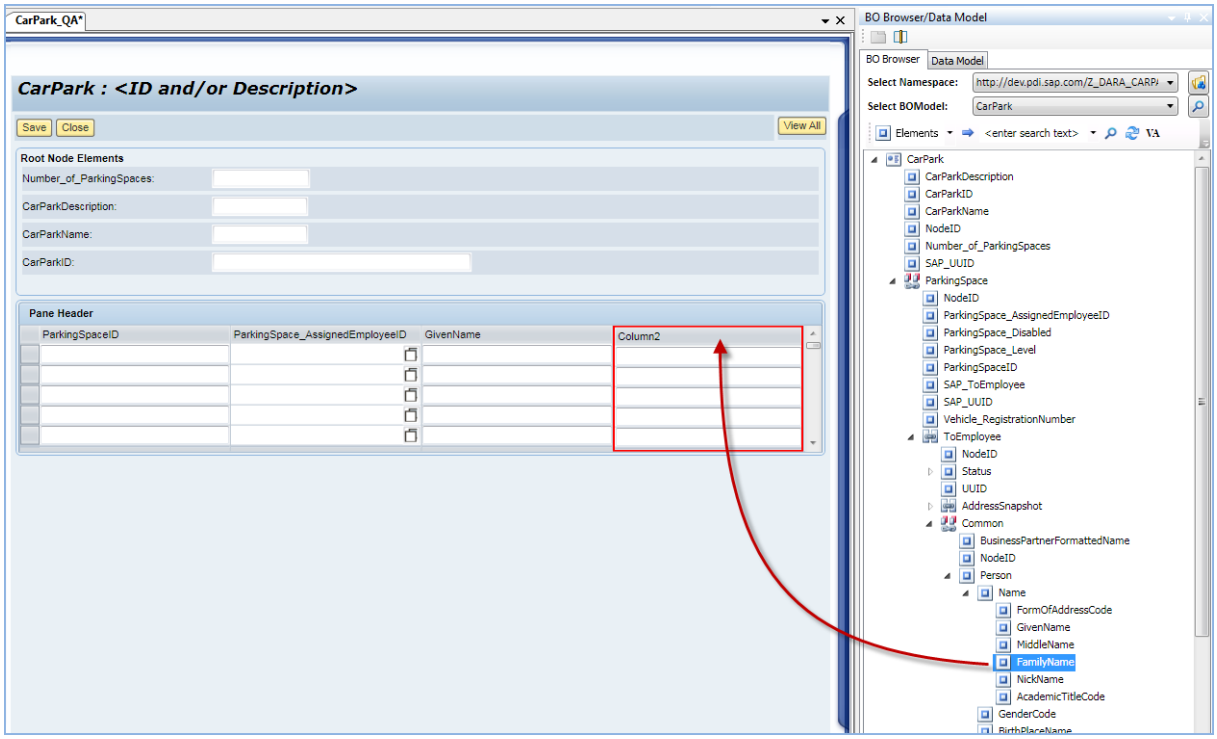
2. Under *ValueHelp*, set the *OVSComponent* to `/SAP_BYD_APPLICATION_UI/publicovs/employeeid/EmployeeID`: In the *Select OVS Component...* dialog box, click *Browse* to select the component.



3. Click *OK*, then *OK* and save your changes.

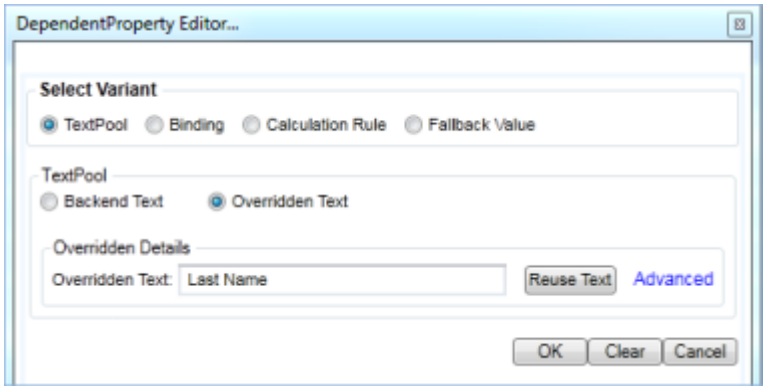
Add the Associated Elements

1. Open the *BO Browser/Data Model*.
2. In the *BO Browser*, expand the *ToEmployee* association. Here you can see the fields of the SAP business object: `Employee`.
3. Select the element *ToEmployee* > *CurrentCommon* > *Person* > *Name* > *GivenName* and drop it onto the third column header.
4. Select the element *ToEmployee* > *CurrentCommon* > *Person* > *Name* > *FamilyName* and drop it onto the fourth column header.
   Both the first and last name will be automatically populated based on the selection of the Employee ID, therefore you do not want these fields to be editable.
5. Select the *GivenName* column and in the Property Explorer, select *Properties* > *Appearance* > *DisplayType* > *StaticText*. Repeat the steps for the *FamilyName* column and save your changes.



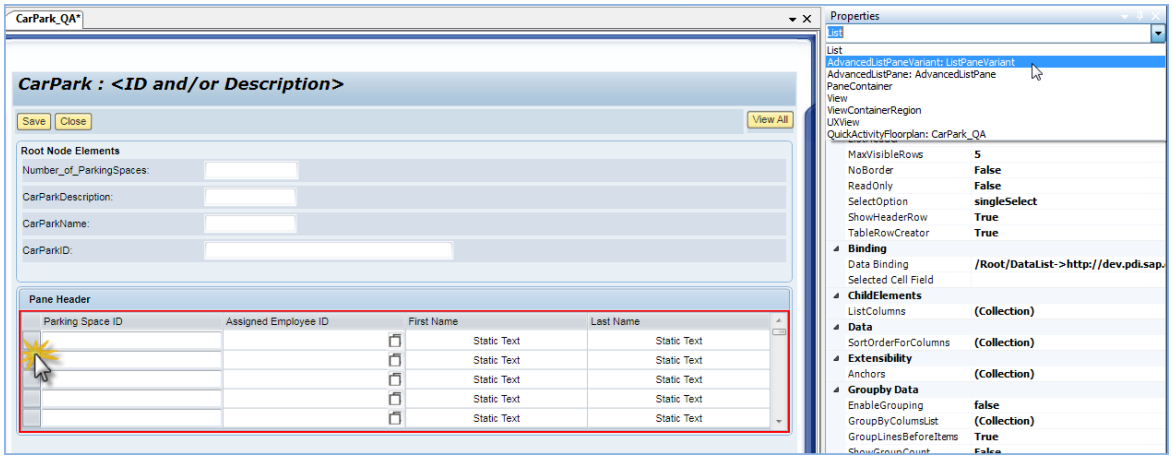
Edit the Column Headers

1. For each column, edit the header *Label* in the *Property Explorer*. Suggested headings are as follows:
   - Parking Space ID
   - Assigned Employee ID
   - First Name
   - Last Name
2. Select one of the columns under *Text Information*.
3. Select the *Label* field and click the down arrow key.
4. The *DependentProperty Editor* opens. In the *Overridden Text* field, change the text and click *OK*. Save your changes.



Add a Toolbar

1. Select the list pane by clicking the area to the left of the first column.
2. In the *Property Explorer* select *AdvancedListPaneVariant: ListPaneVariant*.



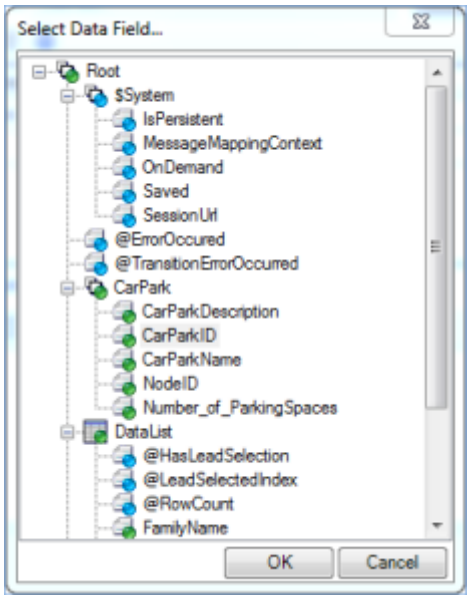
3. Under *Toolbar Information*, set the *UseToolbar* value to *True*.
   By default, you get an *Add Row* button and a *Remove* button. However, you have to add and configure an event handler for each button.
4. Click the *Add Row* button.
5. In the *Property Explorer*, under *Events*, select *OnClick* > *...New Event Handler...* .
   The *Configure EventHandler* dialog box opens.
6. Enter a name for the event handler (for example **`AddRowHandler`**).
7. Add an operation of type *List*:
   1. Click *Add Row*.
   2. In the *Type* field, select *List*.
8. Set the *Operation* to *AddRow* and the *Target List* to */Root/DataList*. Click *OK*.
9. Repeat the above steps for the Remove button, but this time give it the name `RemoveRowHandler` and select `RemoveRow` as the *Operation*.
10. Add an additional operation of type *WindowAction* and select *Save* as the *Action Type*. This ensures that a save is performed after a row is deleted. Click *OK*.
11. Save your changes.

Change the Floorplan Title

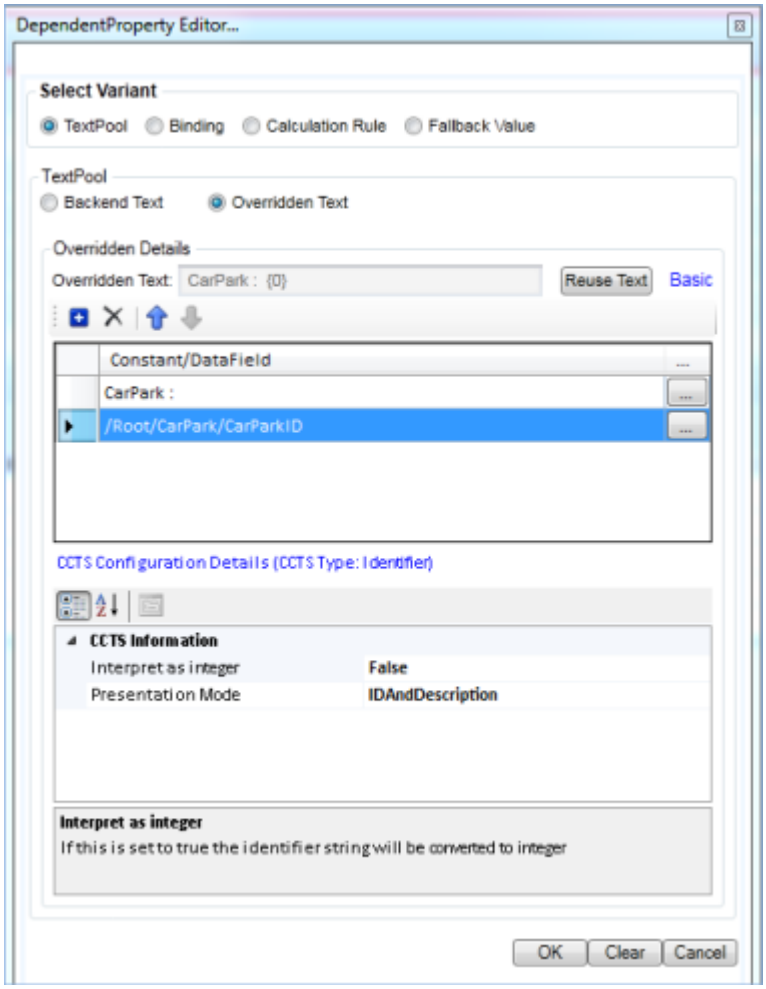
You can create a variable title, so that the title reads "Car Park X" – where "X" is the car park ID.

1. On the *Designer* tab, select the *Identification Region*.
2. In the *Property Explorer*, under *Misc* select *Floorplan Title*.
3. In the *DependentProperty Editor*, select *Overridden Text*.
4. Replace the existing *Constant/DataField* with the text **`Car Park:`** , including the blank space at the end. Click *Advanced*.
5. Click the plus button to create a new string or data field.

6. Click the three dots at the end of the new row.
7. In the *Select Data Field* dialog box, select the data field *CarParkID*.



8. Click *OK*.



9. Click *OK* and save your changes.

Change the Title of the List Pane

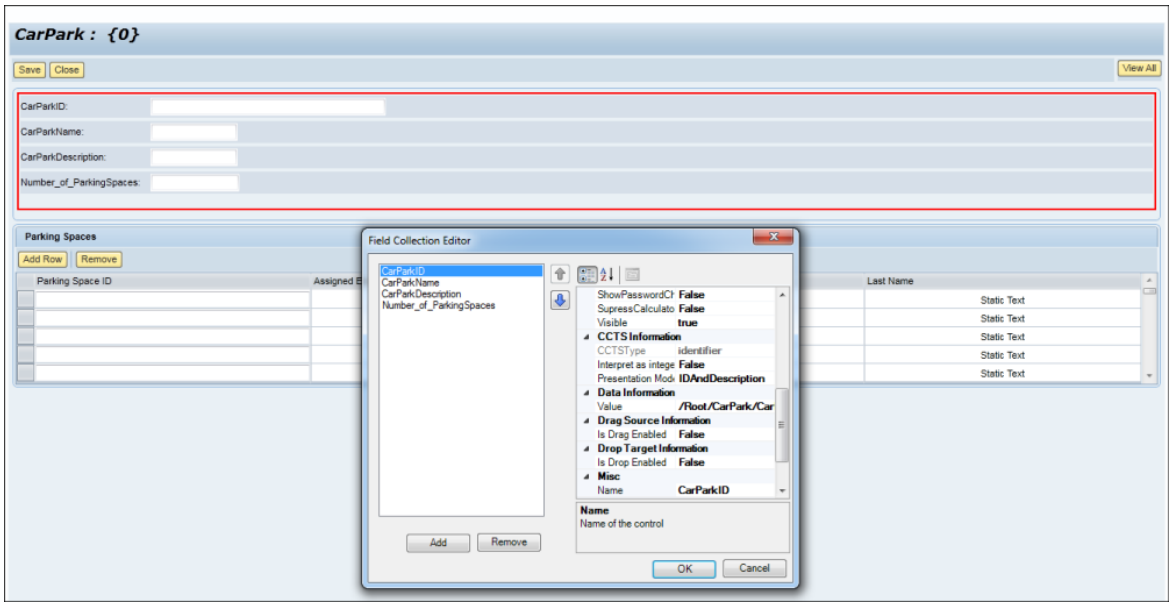
1. On the *Designer* tab, select the list pane.
2. In the *Property Explorer*, under *Pane Header* select *Pane Header*.
3. In the *DependentProperty Editor*, select *Overridden Text* and enter a name for the pane (for example `Parking Spaces`).

Remove the Title of the Form Pane

1. On the *Designer* tab, click the text *Root Node Elements* to select the *Section Group* within the form pane.
2. In the *Property Explorer*, under *Text Information* set *Show Header* to *False*.

Change the Order of the Fields in the Form Pane

1. Select the Section Group.
2. In the *Property Explorer*, select *Fields* under *Contents*.
   The *Field Collection Editor* dialog box is displayed.
3. Select a field and click the arrows to move it up or down the order.

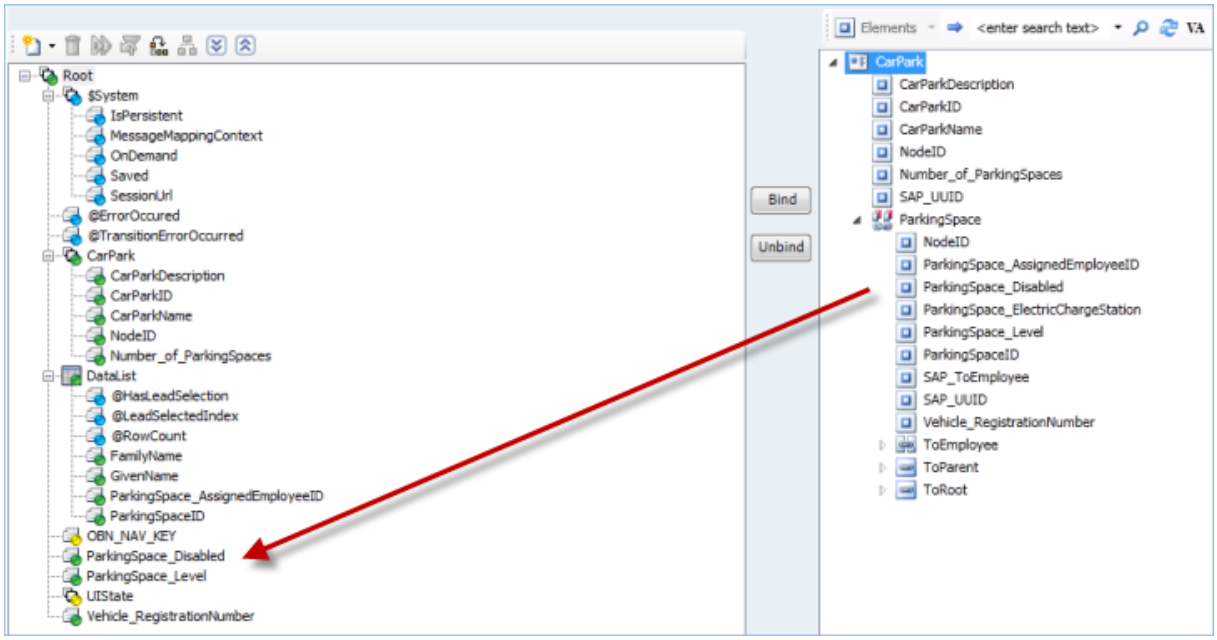


Edit the Field Labels in the Form Pane

1. To edit the name of a field, select the field and edit the *Label* in the *Property Explorer*.
2. Save your changes.

Add Details Area

1. On the *Designer* tab, select the *Parking Spaces* section and click *Add FlexLayoutRow*.
2. Open the BO Browser/Data Model and drag and drop the following elements from the BO Browser onto the new row:
   - ParkingSpace_Disabled
   - ParkingSpace_Level
   - Vehicle_RegistrationNumber
3. For each field, edit the *Label* in the *Property Explorer* (Disabled Friendly Parking, Parking Level, Vehicle Registration Number).
4. Select the *Section Group* and change the *Field Group Header* text in the *Property Explorer* to `Details`.
5. Open the *DataModel* tab and drag the newly added fields into the *DataList* on the left. This links these fields to the individual parking spaces.
6. Click *Save and Activate* and close the UI designer.



Edit the Work Center Title

The default work center title is taken from the business object name. You can change this title by editing the work center floorplan (WCF).

1. In the *Solution Explorer*, double-click the `WCF` screen to open it in the UI designer.
2. On the *Designer* tab, click the folder icon to select the *OberonCenterStructure*.
3. In the *Property Explorer*, click the *Title* field.
4. In the *DependentProperty Editor*, select *Overridden Text* and enter a name for the work center (for example **`Car Park Manager`**).

> **i Note**
>
> If other users have completed this example, there may be multiple work centers with the same title. In this case, you may want to give the work center a unique title, such as **`<your name> – Car Park Manager`**.

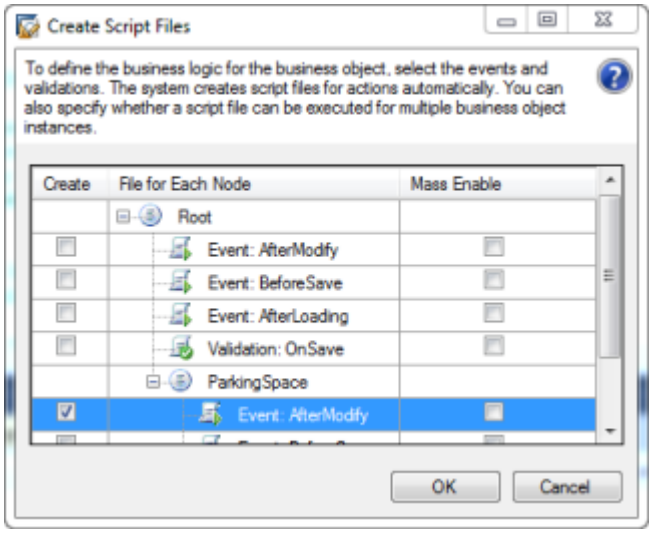
5. Click *OK*, then *Save and Activate* and close the UI designer.

Create the AfterModify Script File

You have set up the QAF screen and created the OVS to find the Employee ID. However, you need a script file to assign the value selected in the OVS to the `Parkingspace_AssignedEmployeeID` element and to update the other fields (first name, last name, etc.) after this selection.

1. In the *Solution Explorer*, right-click the business object and select *Create Script Files*.
2. In the *Create Script Files* dialog box, create an AfterModify script file for the ParkingSpace node.

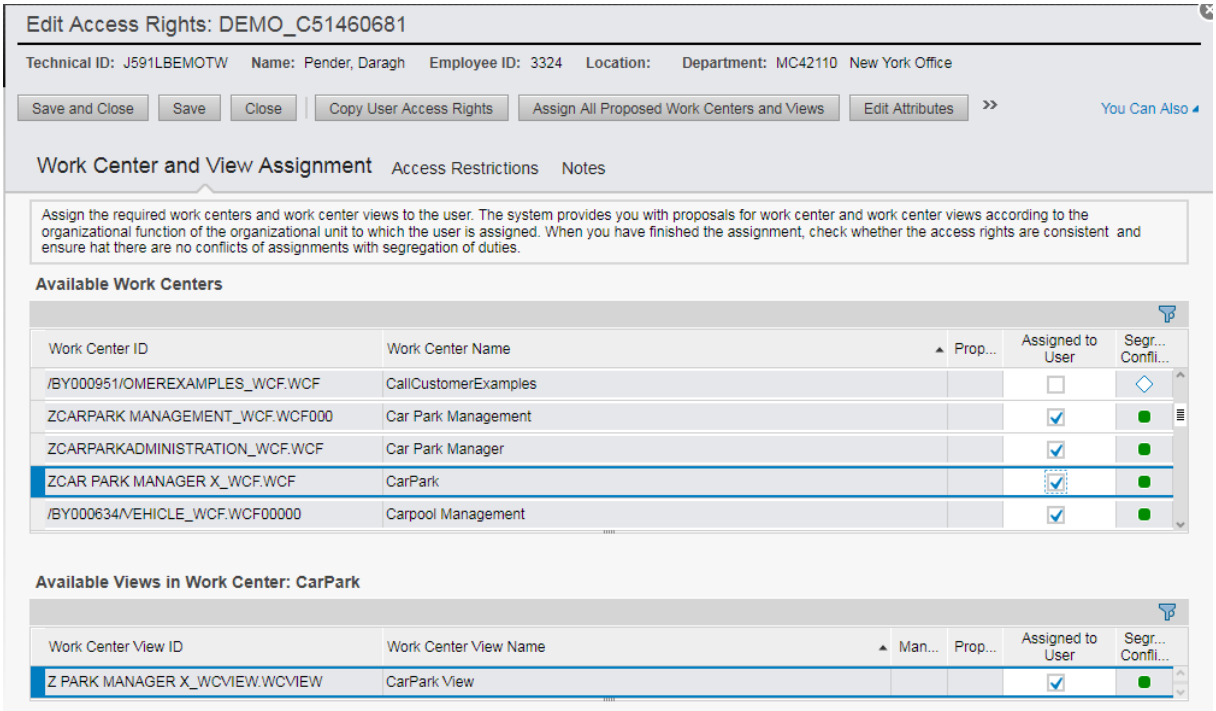
> **i Note**
>
> The *Mass Enable* checkboxes are selected by default.



3. Enter the script code from here [page 115].
4. Save and activate the script file.

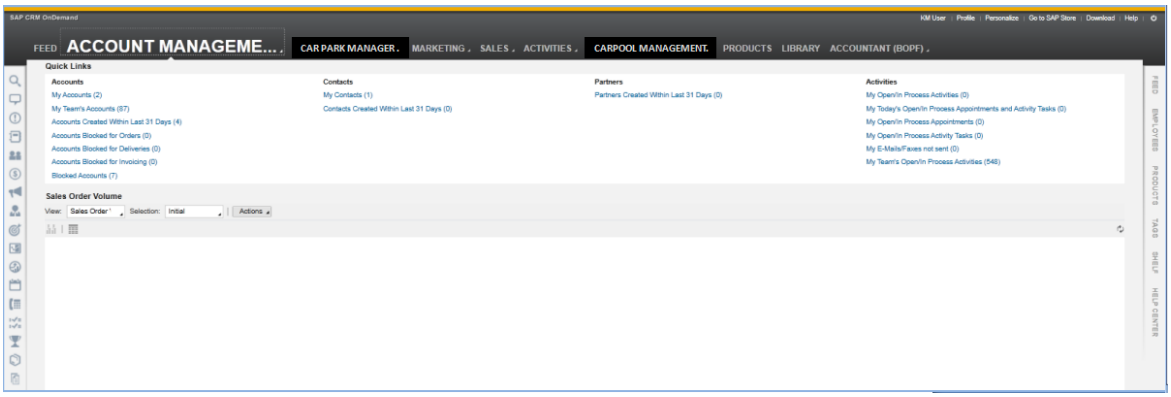
Assign Work Center

Assign rights to your business user so that you can access the new work center in the SAP cloud solution. For more information, see User Setup Quick Guide [page 33].

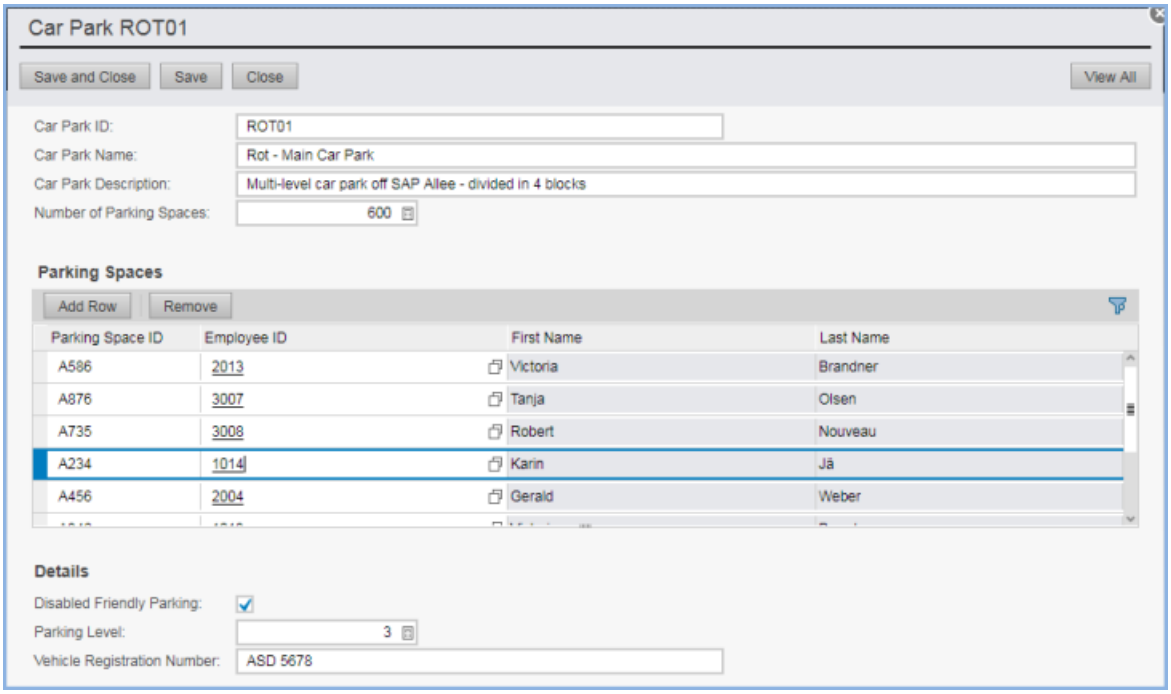


Test the Solution

1. Log on to the SAP cloud solution with a user that has access to the *Car Park Manager* work center.
2. Click the *Car Park Manager* work center to open the *Car Park* OWL.



From the OWL you can create new car parks or edit the details of existing car parks.



3. Select a car park and click *Edit* to bring up the QAF.
4. Click *Add Row* to create a new parking space and enter a *Parking Space ID*.
5. Click the icon in the *Employee ID* field to bring up the OVS and select an employee.
   The *First Name* and *Last Name* fields are automatically populated.
6. Enter additional information in the *Details* area.

Further Development

Now that you have developed a solution, you can begin to appreciate the power and flexibility of the studio.

There are many enhancements you could make to this solution. Here are just a few ideas:

- Auto-refresh the OWL after creating a new car park. For more information, see here [page 119].
- Validate the car park on save, so that you cannot assign more than the total number of car spaces. An error message is displayed to the user if they attempt to save a car park that fails this validation rule. For more information, see here [page 138].
- Add a field to calculate the number of unassigned car spaces. For more information, see here [page 120].
- Create a read-only car park fact sheet. For more information, see here [page 124].
  - Create a View button on the OWL that links to the carpark fact sheet using object-based navigation (OBN). For more information, see here [page 129].
  - Create a link from the car park fact sheet to the employee fact sheet using object-based navigation (OBN). For more information, see here [page 132].
- Extend the employee business object to show the Parking Space ID on the employee fact sheet. For more information, see here [page 136].
- Add a URL mashup to display the car park location on a map. For more information, see here [page 116].
- Add a form to provide a PDF printout of parking spaces and assigned employees.

- Add date/time information to each parking space. You could then add a cost per day or per hour.

5.8.2 Sample Code

5.8.2.1 Sample Code: Business Object CarPark

```
import AP.Common.GDT as apCommonGDT;
import AP.FO.BusinessPartner.Global;
businessobject CarPark
  // *Further Development*
    // declare an error message that is called if the BO fails validation
    // raises Message_Car_Park_Full

    {

    // Root node must not be specified, it is created implicitly
    //
    // Add elements here.
    //
    // To declare an element as alternative key, use the [AlternativeKey]
annotation.
    // Please note:
    // - An alternative key of type UUID exists implicitly for each node,
enabling the node
    //   to be used as an association target.
    // - The QueryByElements query is created automatically for each node.
    // - Code completion is available with CTRL + SPACE.

    // *Further Development*
    // define an error message to be displayed if the BO fails validation
    // message Message_Car_Park_Full text "The car park is full - no spaces
available";

    // Car Park ID - (the company may have multiple car parks)
    // - therefore we use this ID to identify the individual car park
    element CarParkID : ID;

    // additional information on the car park
    element CarParkName : LANGUAGEINDEPENDENT_LONG_Text;
    element CarParkDescription : LANGUAGEINDEPENDENT_LONG_Description;
    element Number_of_ParkingSpaces : IntegerValue;

    // *Further Development*
    // add an element to display how many parking spaces are available
    // element Number_of_ParkingSpaces_Available : IntegerValue;
    // declare an action to calculate the available car spaces
    // action Update_AvailableParkingSpaces;

    // *Further Development*
    // add an element to record location information
    // this field is used with a URL mashup to display the location of the car
park
    // element CarParkLocation : LANGUAGEINDEPENDENT_MEDIUM_Description;


    // each car park will have multiple parking spaces
    // we create a node with 0 to n parking spaces

    node ParkingSpace [0,n]    {
```

```
      // each parking space is assigned a unique identifier
      element ParkingSpaceID : ID;

      // display the ID of the Employee assigned to the parking space
      element ParkingSpace_AssignedEmployeeID : ID;

      // a parking space may be designated as a "Disabled Friendly" parking space
      element ParkingSpace_Disabled : Indicator;

      // is there an electric charging station for this parking space ?
      element ParkingSpace_ElectricChargeStation : Indicator;

      // display the license plate number of the employee's car
      element Vehicle_RegistrationNumber : LANGUAGEINDEPENDENT_MEDIUM_Text;

      // for a multi-level car park, we have the possibility to specify a level
number
      element ParkingSpace_Level : NumberValue;

      // we need to create an association to the SAP BO "Employee"
      // this allows us to include elements from the Employee BO in our screens
      // a parking space may be associated to at most one employee
      association ToEmployee [0,1] to Employee;

      }
}
```

5.8.2.2 Sample Code: Event-AfterModify CarPark-ParkingSpace

```
import  ABSL;
import  AP.FO.BusinessPartner.Global;
// Auxiliary flag that will help us decide whether we have to
// update the ToEmployee association.
var  bUpdateEmployee   = false;
// Our pointer to the employees' query.
var  qryEmployees;
// Object that will hold the parameters for our employees' query call.
var  paramEmployee;
// Collection of result values of our employees' query call.
var  resultEmployees;
// Check whether our ToEmployee association is still up to date. If not, we'll
// have to update it.
if  (this.ParkingSpace_AssignedEmployeeID.IsInitial()) {
  // => No employee indicated for the parking space.
  if  (this.ToEmployee.IsSet()) {
    // => Our ToEmployee association still holds the employee that had been
    //    associated before - so now we have to reset the association.
    this.ToEmployee.Reset();
  }
}
else if (!this.ToEmployee.IsSet()) {
  // => We do have an employee indicated for the parking space, however,
  //    the association ToEmployee hasn't been initialized, yet.
  // => Set our flag to true.
  bUpdateEmployee = true;
}
else if (this.ToEmployee.IdentificationEmployeeID.EmployeeID.content !=
this.ParkingSpace_AssignedEmployeeID) {
  // => We do have an employee indicated for the parking space, however,
  //    the association ToEmployee still holds the employee that had been
associated to the space before.
```

```
  // => Set our flag to true.
  bUpdateEmployee = true;
}
if (bUpdateEmployee) {
  // => Our association to the employee needs to be refreshed.
  // Reset it first, since the new employee ID might be invalid.
  this.ToEmployee.Reset();
  // Now look for the details of the given employee:
   // ...connect to the BO Employee's query...
  qryEmployees = Employee.Identification.QueryByEmployeeAttributes;
  // ...populate the parameters collection for the query call...
  paramEmployee = qryEmployees.CreateSelectionParams();
  // ...saying that the ID of the employee we're looking for must be the ID
indicated by our user...
  paramEmployee.Add(qryEmployees.EmployeeID.content, "I", "EQ",
this.ParkingSpace_AssignedEmployeeID);
  // ...and execute the query!
  resultEmployees = qryEmployees.Execute(paramEmployee);
  // Hopefully we've found an employee to the given ID!
  foreach (var  oneEmployee  in  resultEmployees) {
    // Since we know that we can have only one employee to the ID, we
    // take the first result and exit the loop.
    this.ToEmployee  = oneEmployee.ToRoot;
    break;

  }
}
```

5.8.3 Further Development

5.8.3.1 Add a URL Mashup to an SAP Floorplan

Overview

In this example you are going to add a URL mashup to the quick activity (QA) floorplan. This mashup will display the car park location on a map, based on the location information entered.

Prerequisites

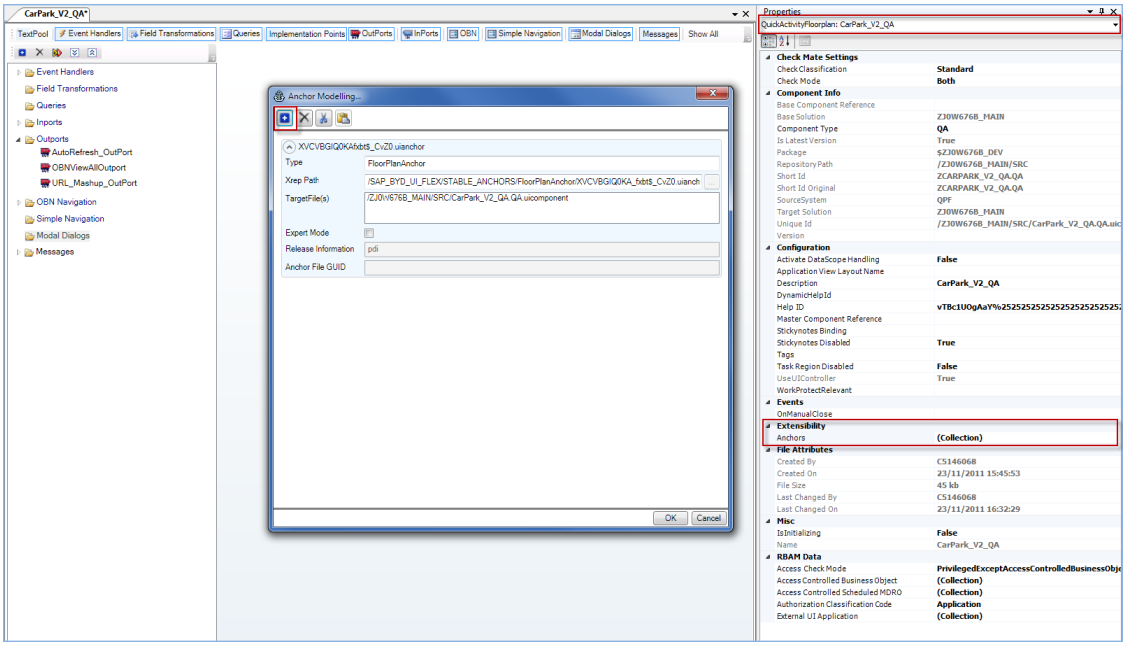
- You have completed the example: Building a Solution [page 99]
- The mashup has been created in the SAP cloud solution. For more information, see Mashups Quick Guide [page 551].

Procedure

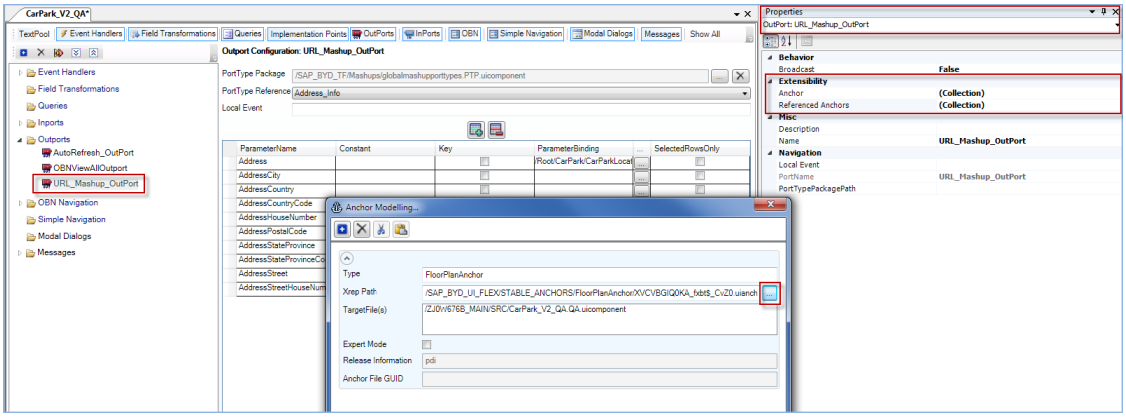
1. Add an element on the root level of the Car Park business object to record location information.

```
element CarParkLocation : LANGUAGEINDEPENDENT_Text;
```

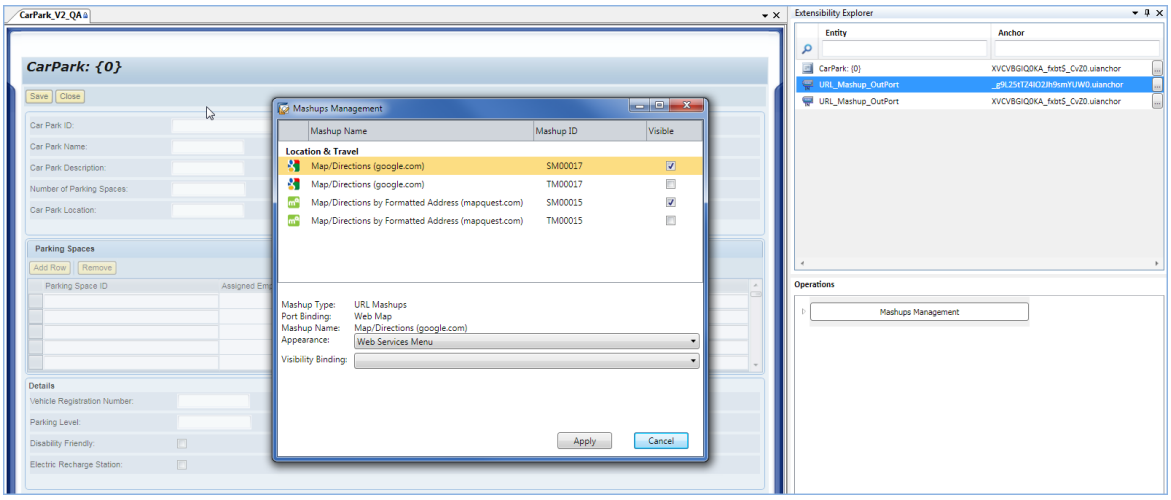
2. Save and activate the business object.
3. Double-click the quick activity floorplan to open it in the UI designer.
4. Click the *Update Metadata* button to update the BO metadata.
5. Drag the *CarParkLocation* from the *BO Browser / Data Model* onto the *Designer* tab.
6. Change the label for the field to **`Car Park Location`**.
7. On the *Controller* tab, perform the following actions:
    1. Add a new outport with the name **`URL_Mashup_OutPort`** and the following settings:
        - *PortType Package* = `/SAP_BYD_TF/Mashups/globalmashupsupporttypes`
        - *PortType Reference* = `Address_Info`
    2. Bind the *Address* parameter to the *CarParkLocation* data field.
    3. In the *Property Explorer*, select the QA floorplan from the drop-down list at the top.
    4. Click the *Anchor* field under *Extensibility*.
    The *Anchor Modelling* dialog box appears.
    5. Click the plus sign to create a stable anchor for the floorplan.



    6. In the *Property Explorer*, select the mashup outport from the drop-down list at the top.
    7. Click the *Anchor* field under *Extensibility*.
    The *Anchor Modelling* dialog box appears.
    8. Click the plus sign to create a stable anchor for the floorplan.
    9. In the *Property Explorer*, click the *Referenced Anchor* field under *Extensibility*.
    The *Anchor Modelling* dialog box appears.
    10. Click the plus sign to create a referenced anchor for the outport.
    11. Click the three dots next to the *XRep Path* field.
    12. In the *Configured Anchors* dialog box, click the anchor you created for the QA floorplan.



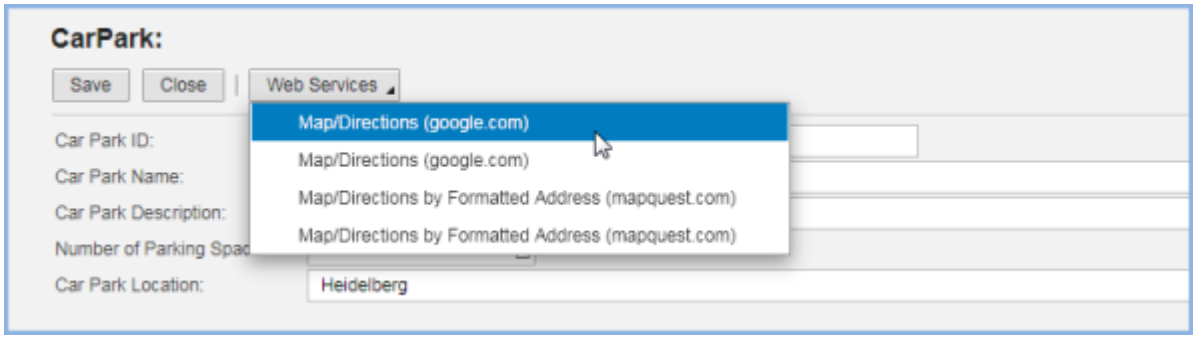
8. Save and activate.
9. Click the *Designer* tab.
10. In the *Extensibility Explorer*, select the mashup outport and click the *Mashups Management* button.
    The *Mashups Management* dialog box appears, listing the available mashups.
11. Select the mashup you want to use and click *Apply*.

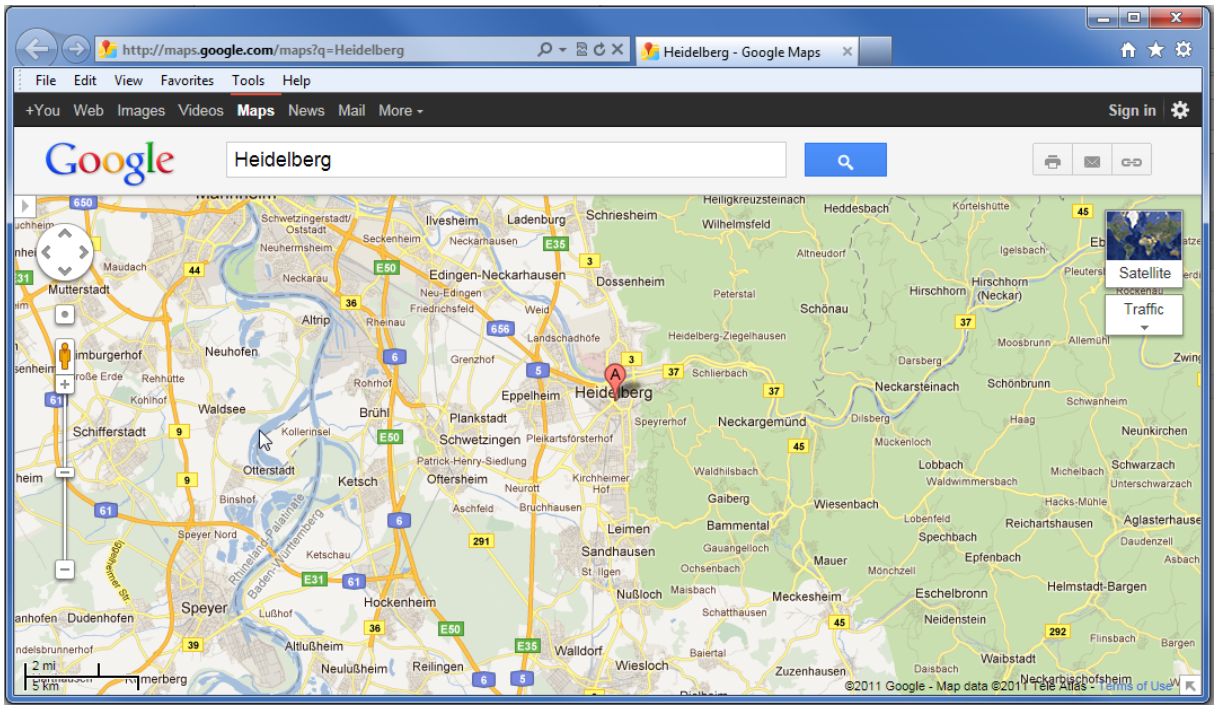


12. Save and activate.

Result

The mashup is available from the *Web Services* menu on the QA. When you click the mashup, the URL is opened with the value entered into the *Car Park Location* field as an input parameter.





5.8.3.2 Configure the OWL to Auto-Refresh

In this example you are going to configure the car park object work list (OWL) to refresh automatically when a new car park instance is saved on the quick activity floorplan (QAF).

Prerequisite

You have completed the example: Building a Solution [page 99]

Procedure

1. Open the object work list (OWL) in the UI designer.
2. On the *Controller* tab, create a new *Inport* with the name **`AutoRefresh_Inport`**.
3. In the *Property Explorer*, perform the following actions:
   - set the property *Events* > *OnFire* to the default *EventHandler* – this event handler executes the query for the OWL
   - enter the name **`AutoRefresh_LocalEvent`** in the field *Navigation* > *Local Event*.
4. Save and activate.
5. Open the quick activity floorplan (QAF) in the UI designer.
6. On the *Controller* tab, create a new *Outport* with the name **`AutoRefresh_Outport`**.
7. In the *Property Explorer*, perform the following actions:
   - set the property *Behavior* > *Broadcast* to *True*.
   - enter the name **`AutoRefresh_LocalEvent`** in the field *Navigation* > *Local Event*.
8. On the *Controller* tab, edit the *SaveHandler* to include a new operation of type *FireOutport* below the existing *WindowAction* operation.

   > **i** Note
   >
   > Operations are executed from top to bottom, so you need to save the new business object instance before running the default query.

9. Save and activate

Result

When you save a new car park instance on the QAF, then return to the OWL, the list of car parks is updated with the car park that you created - you do not have to click *Go* to run the query again.

5.8.3.3 Calculate the Number of Available Car Parking Spaces

In this example you are going to add a field that automatically calculates the remaining car parking spaces available, based on the total number of spaces and the number of spaces assigned to employees. You will use an action to calculate the field value and you will call this action from a BeforeSave event on the root node. The result is displayed as a column on the object work list.

Prerequisite

You have completed the example: Building a Solution [page 99]

Procedure

1. In the studio, open the Car Park business object.
2. In the root node, add an element of type `IntegerValue` with the name **`AvailableParkingSpaces`**.

```
element AvailableParkingSpaces : IntegerValue;
```

3. In the root node, declare an action called **`Update_AvailableParkingSpaces`**.

```
action Update_AvailableParkingSpaces;
```

4. Save and activate the business object.
5. Right-click the Car Park business object and select *Create Script Files*.
6. Create the *Action-Update_AvailableParkingSpaces* and also the *BeforeSave* event for the root node.
7. In the action script file, enter the code to calculate the number of available car parking spaces:

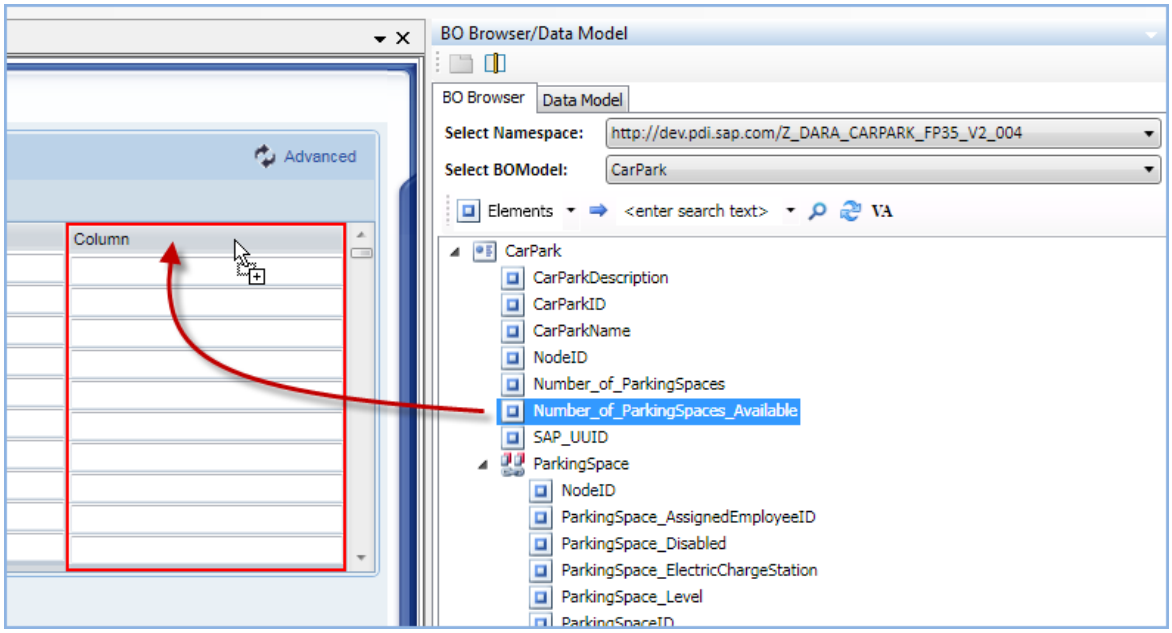
```
this.AvailableParkingSpaces = this.Number_of_ParkingSpaces-
this.ParkingSpace.Count();
```

8. Save and activate the action.
9. In the *BeforeSave* event, enter the code to call the *Action-Update_AvailableParkingSpaces*.

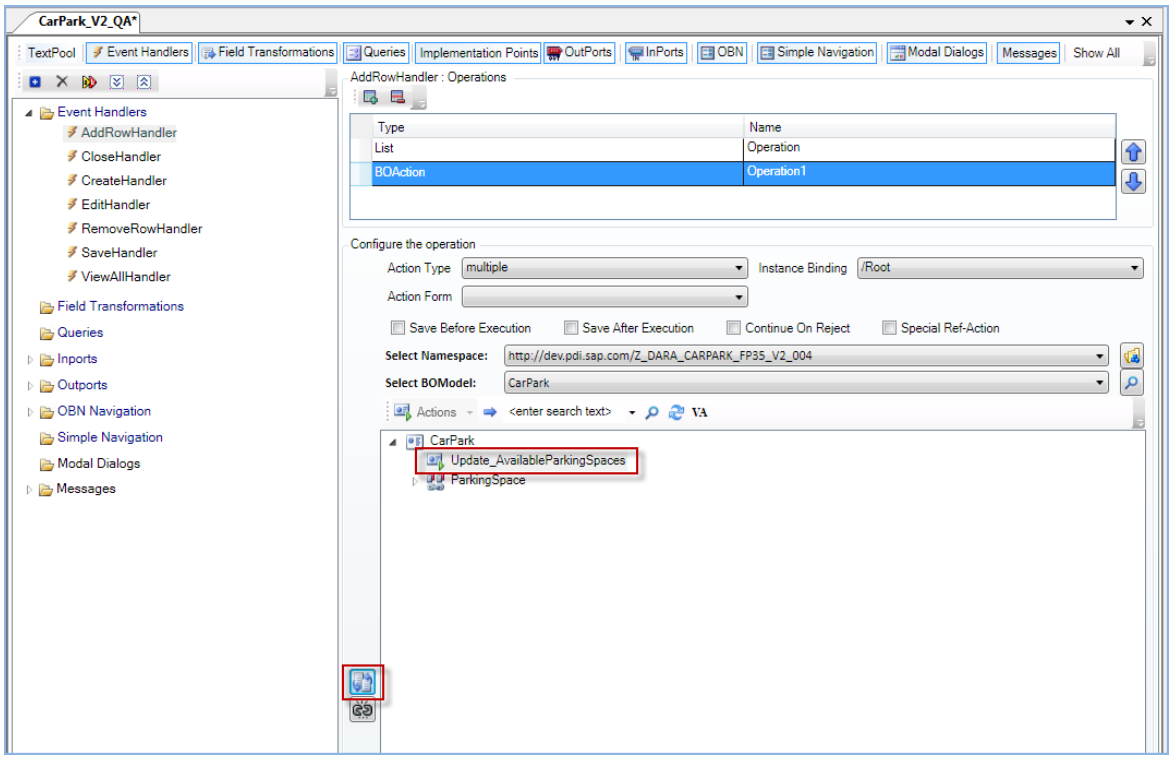
```
this.Update_AvailableParkingSpaces();
```

> **i Note**
>
> This will calculate the available car spaces whenever the Car Park BO is changed at root level and saved. It will not run the script if a car space is added or removed. To ensure that the script is run whenever a car space is added or removed, later in this procedure, you will edit both the *AddRowHandler* and the *RemoveRowHandler* to call the *Update_AvailableParkingSpaces* action.

10. Save and activate the event.
11. Open the object work list in the UI designer.
12. On the *Designer* tab, add a column on the right of the table.
13. Drag the element *Number_of_ParkingSpaces_Available* from the *BO Browser/Data Model* onto the column header.

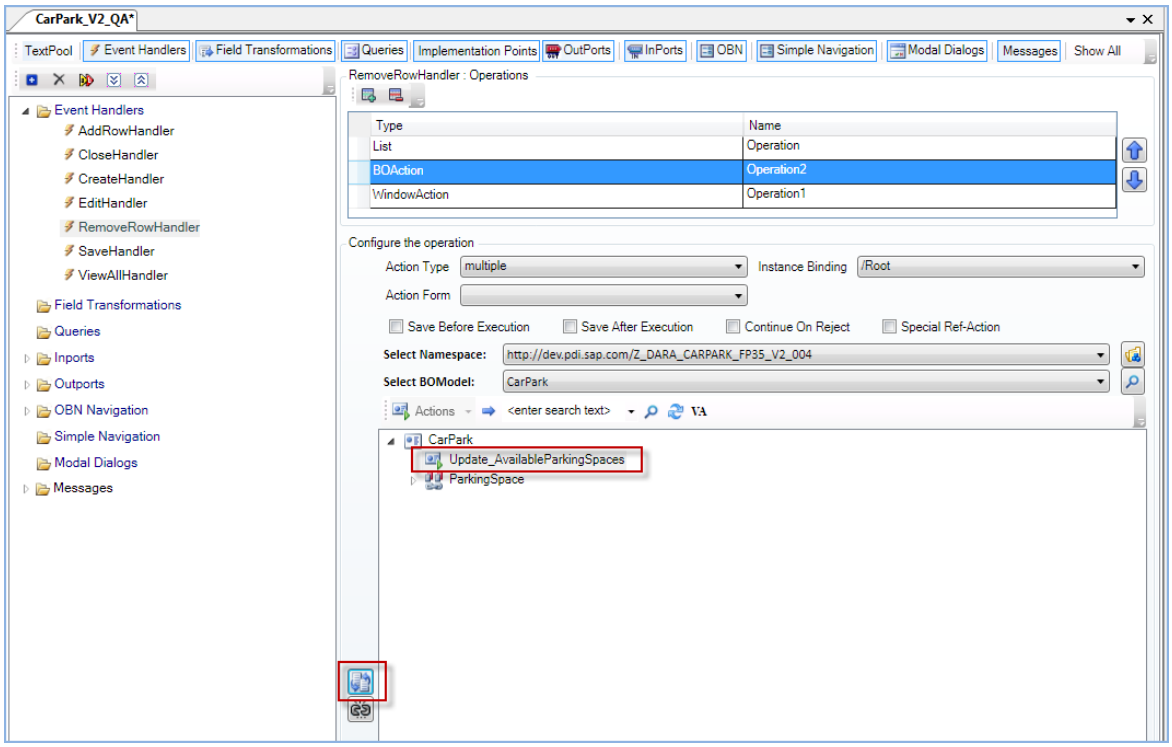


14. Select not to use backend text when prompted.
15. Edit the *Label* for the column.
16. Save and activate the floorplan.
17. Open the QA floorplan in the UI designer.
18. On the *Controller* tab, select the *AddRowHandler*.
19. Add a new operation of type *BO Action*.
20. Select the *Update_AvailableParkingSpaces* action and click the *Bind* button.



21. Select the *RemoveRowHandler*.

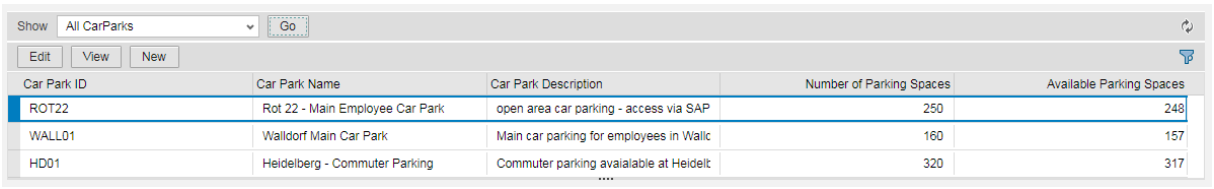
22. Add a new operation of type *BO Action* before the *WindowAction*.
23. Select the *Update_AvailableParkingSpaces* action and click the *Bind* button.



24. Save and activate.

Result

The available parking spaces are now displayed for each car park. The number will be updated if you edit a car park and change the total number of car spaces, add a new space, or delete a space. You have to click *Go* on the OWL to re-run the query and display the current information.



5.8.3.4 Create a Fact Sheet for the Car Park

Overview

In this example you are going to create a fact sheet for a car park instance. The fact sheet (FS) provides a read-only version of the information entered on the quick activity (QA) floorplan. The main steps in creating a fact sheet are as follows:

- create the floorplan layout
- configure the floorplan to allow for object-based navigation from other floorplans to this floorplan:
  - add the fields `OBN_NAV_KEY` and `NodeID` to the data model
  - create the inport
  - create the event handler for the inport
- include the FS in the assigned objects of the work center view (WCV)

> **i Note**
>
> It is intended that the fact sheet will be accessed from a *View* button on the object work list (OWL). For more information about creating the button and linking to the car park fact sheet, see here [page 129].

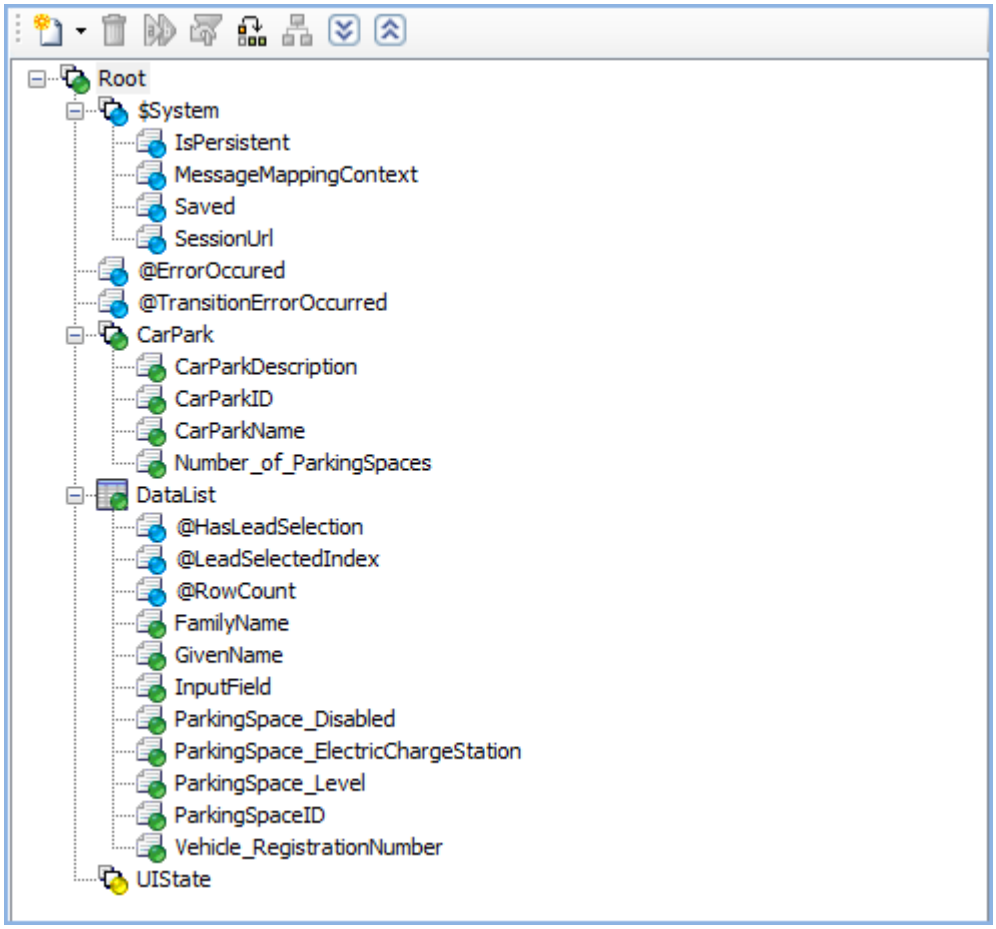
Prerequisites

- You have completed the example: Building a Solution [page 99]

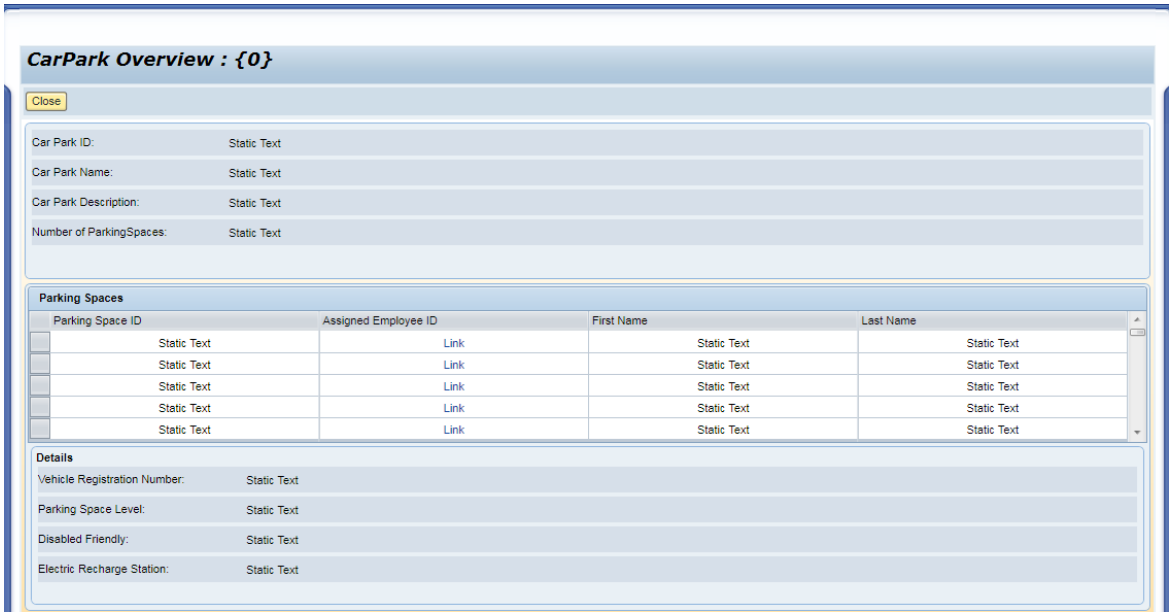
Procedure

1. Right-click the *CarPark* business object and select *Create Screens*.
2. Select the *fact sheet* floorplan and click *OK*.
3. Double-click the fact sheet to open it in the UI designer.
4. Delete the column on the right.
5. Delete the fields *ExIdr*, *ExIdr1* and *ExIdr2*.
6. Delete the *Edit* button.
7. Delete the *View All* button.
8. Select the *IdentificationRegion* and edit the *FloorplanTitle* to include the Car Park ID as a variable.
9. Select the *SectionGroup* and set the property *ShowHeader* to false.
10. Select the *SectionGroup* and select the property *Contents > Fields* to open the *Field Collection Editor* and re-order the fields.
11. Select each field and set the property *DisplayType* to *StaticText*.
12. Select each field and edit the *Label* property.
13. Add a flex layout row below the existing row.
14. From the *Toolbox*, drag-and-drop an *AdvancedListPane* into the new row.
15. Edit the property *PaneHeader* to read *Car Spaces*.

16. Add a column to the table.
17. Drag-and-drop the following fields from the *BO Browser/Data Model* onto the table columns:
    - *ParkingSpaceID*
    - *ParkingSpace_AssignedEmployeeID*
    - *GivenName*
    - *FamilyName*
18. Rename the column header labels:
    - *Parking Space ID*
    - *Assigned Employee ID*
    - *First Name*
    - *Last Name*
19. Set the *DisplayType* of the *Assigned Employee ID* column to *Link*, and set the other columns to *StaticText*.
20. Select the *AdvancedListPane: ListPaneVariant* and set the property *UsePreviewPane* to *True*.
21. Edit the property *PaneHeader* to read *Details*.
22. Drag-and-drop the following fields from the *BO Browser/Data Model* into the *Details* section:
    - *ParkingSpace_Level*
    - *Vehicle_RegistrationNumber*
    - *ParkingSpace_Disabled*
    - *ParkingSpace_ElectricChargeStation*
23. Rename the field labels:
    - *Parking Space Level*
    - *Vehicle Registration Number*
    - *Disabled Friendly*
    - *Electric Recharge Station*
24. Change the order the fields as desired.
25. Optional: On the *DataModel* tab, you can add a data structure element to contain the CarPark root node elements. This makes it easier to view and organize the data model elements, but does not make any functional difference.

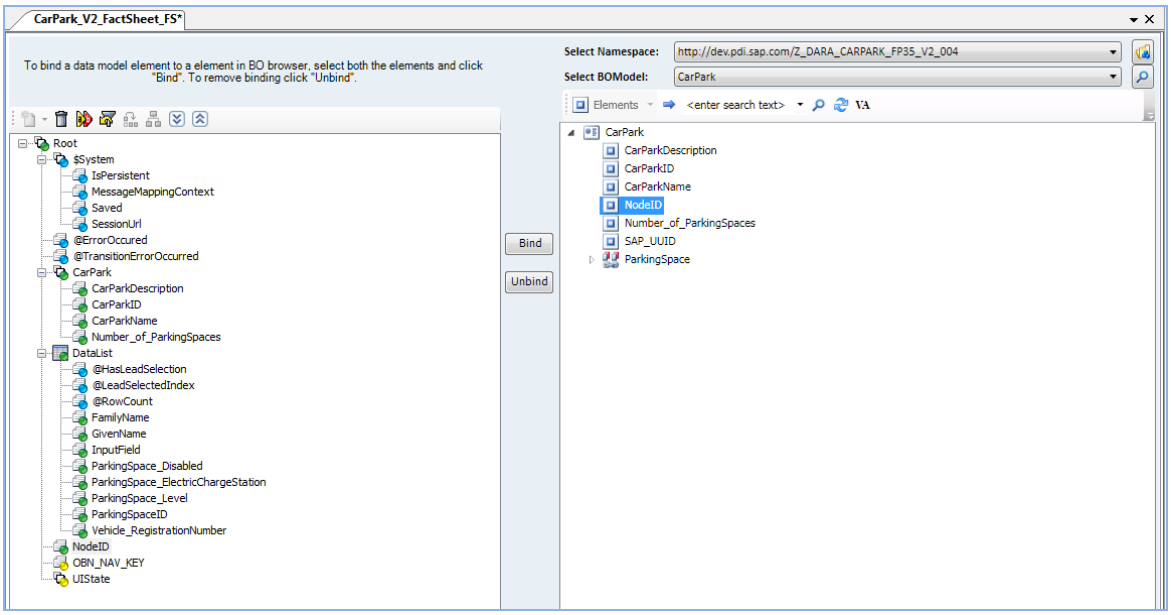


You now have a fact sheet that looks similar to this:

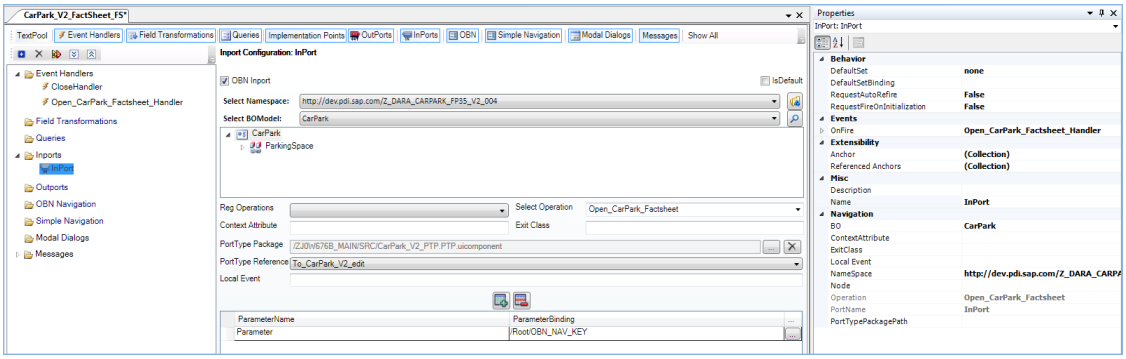


26. On the *DataModel* tab, perform the following actions:
    1. Right-click the *Root* data field and select *Add Data Field*.

2. Right-click the new field, select *Rename* and enter the name **`OBN_NAV_KEY`**.
3. Right-click the *Root* data field and select *Add Data Field*.
4. Select the new data field and select the *NodeID* in the business object model, then click *Bind*.

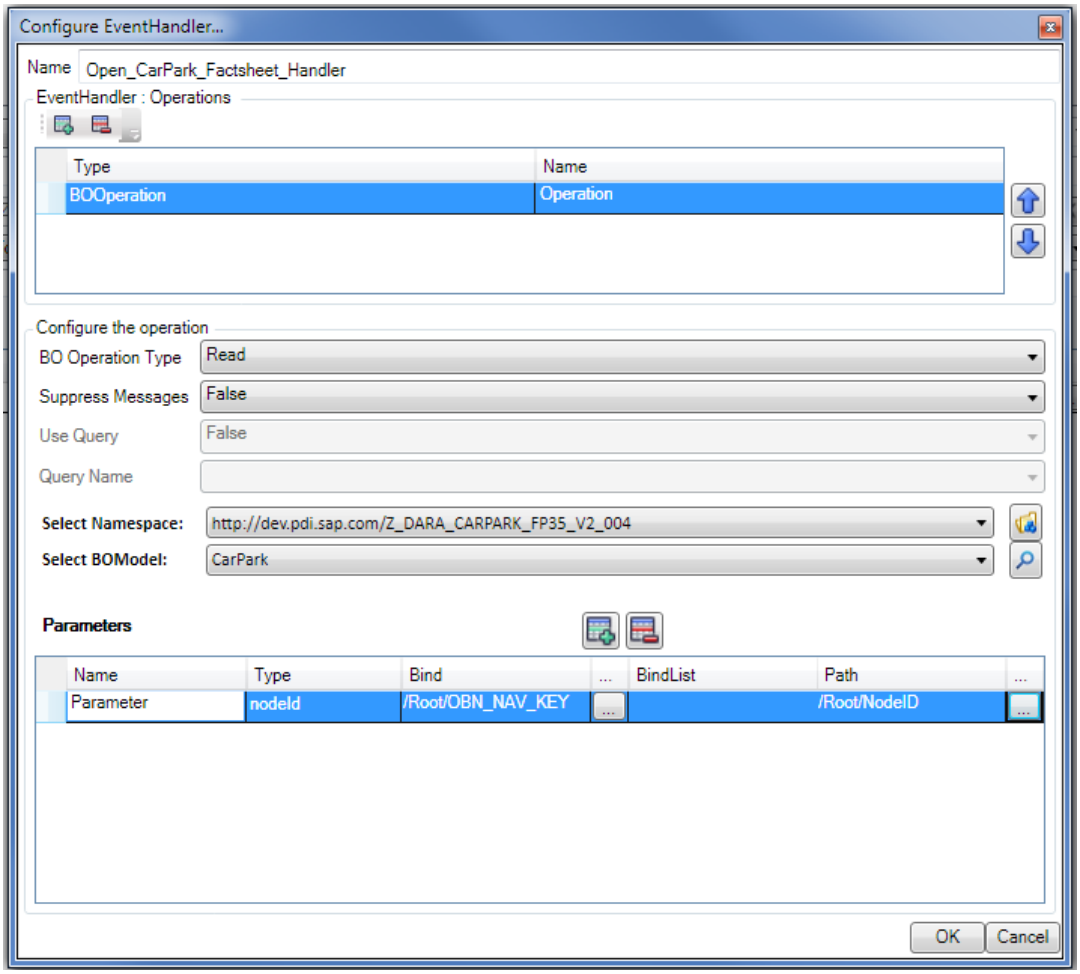


27. On the *Controller* tab, perform the following actions to create the inport:
    1. Right-click *Inports* and select *Add Inport*.
    2. Under *Inport Configuration*, select the check box *OBN Inport*.
    3. Check that the *Namespace* and *BO Model* fields are populated correctly.
    4. In the field *Select Operation*, enter the name **`Open_CarPark_fact sheet`**. This name is used to identify the inport.
    5. Click the three dots next to the *PortType Package* field and select the port type package for the business object. This is located in the folder `<project name>_MAIN/SRC`.
    6. In the *PortType Reference* field, select *To_CarPark_Edit*.
    7. Click the *Add Parameter* button.
    8. Click the three dots in the new *Parameter* row.
    9. In the *Parameter Binding* dialog box, select the field *OBN_NAV_KEY* and click *OK*.



10. In the *Property Explorer*, under *Events*, in the *OnFire* field select *New Event Handler*.
11. In the *Configure Event Handler* dialog box, perform the following actions:
    1. Enter the name **`Open_CarPark_fact sheet_Handler`**.
    2. click the *Add Operation* button.

3. Select *BOOperation* for the operation *Type*.
4. Select *Read* for the *BO Operation Type*.
5. Click the *Add Parameter* button.
6. For the new parameter row, select *nodeId* in the *Type* field.
7. Click the three dots next to the *Bind* field.
8. In the *Parameter Binding* dialog box, select the field *OBN_NAV_KEY* and click *OK*.
9. Click the three dots next to the *Path* field.
10. In the *Parameter Binding* dialog box, select the field *NodeID* and click *OK*.



28. Click *Save and Activate*.
29. In the *Solution Explorer*, double-click the work center view (WCV) to open it in the UI designer.
30. In the *Property Explorer*, under *RBAMData* click *AssignedObjects*.
31. In the *Add Assigned Objects* dialog box, click the *Add* button to create a new row.
32. In the new row, click the three dots next to the *ComponentId* field.
33. In the *Select Component* dialog box, navigate to your project and select the fact sheet then click *OK*.
34. Click *Save and Activate*.

5.8.3.5 Create an OBN to the Car Park Fact Sheet

Overview

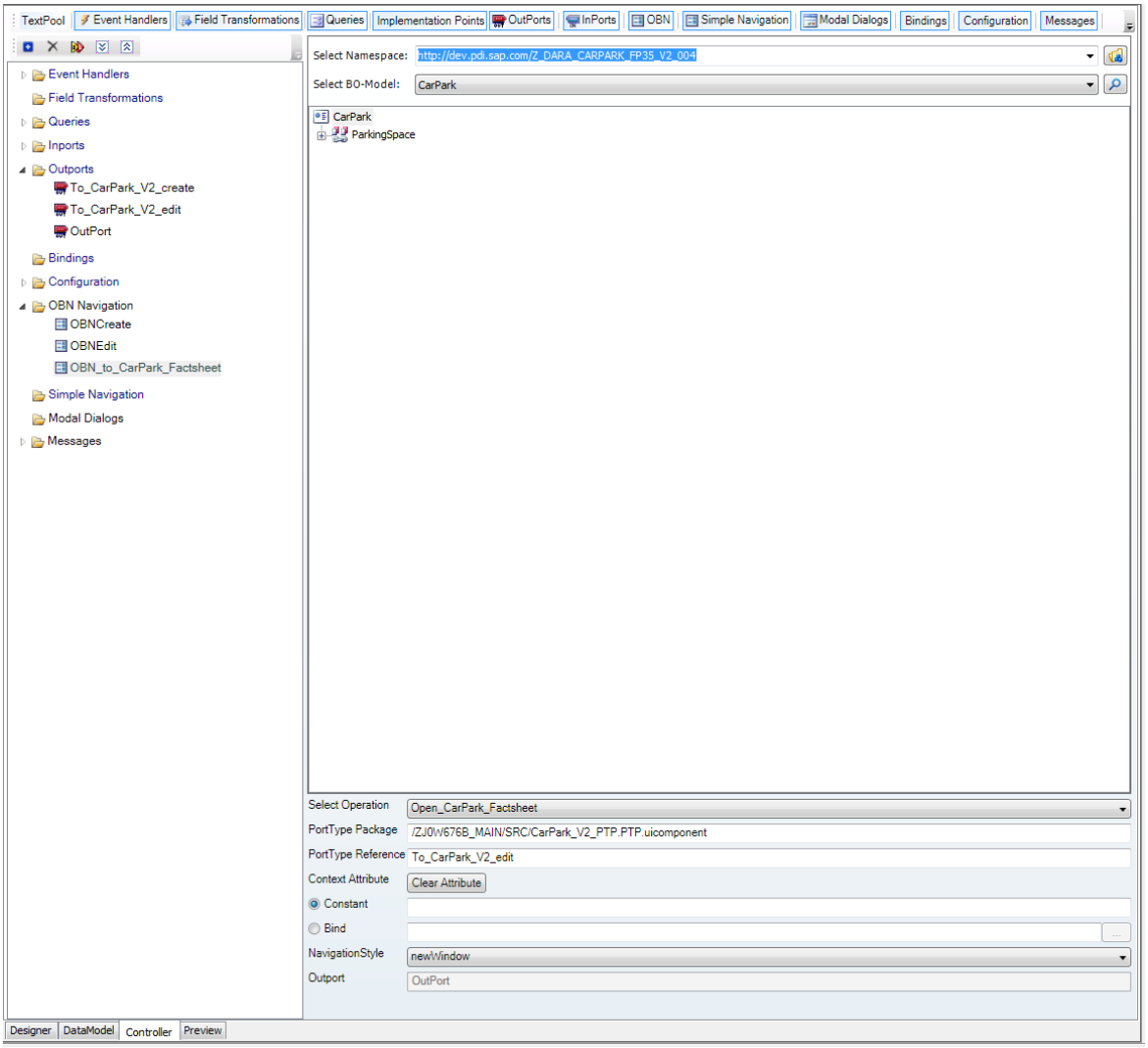
In this example you are going to create an object-based navigation (OBN) link from the Car Park Manager object work list (OWL) to the Car Park fact sheet. You have already configured the fact sheet with an inport that allows for OBN, now you are going to create an OBN outport to connect to this inport.

Prerequisites

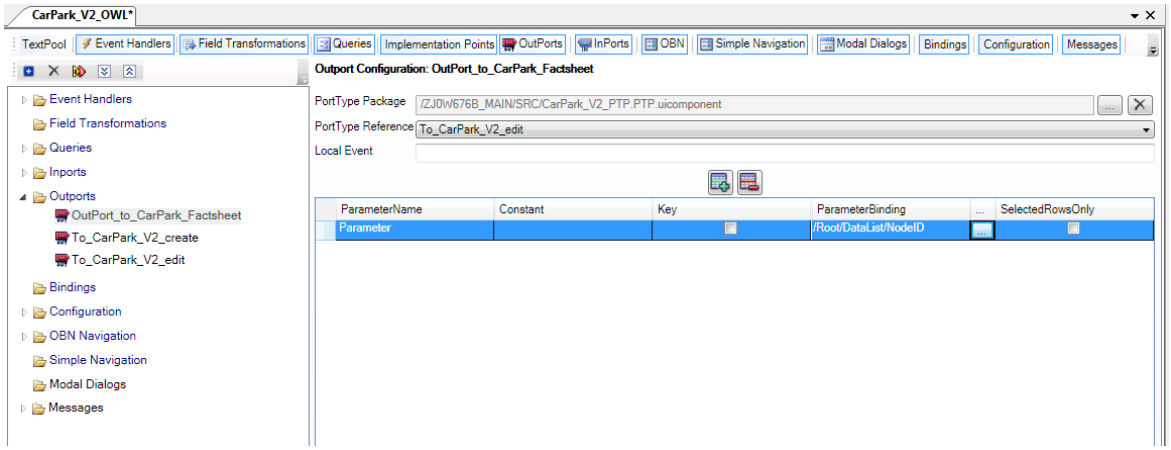
- You have completed the example: Building a Solution [page 99].
- You have completed the example: Create a Fact Sheet for the Car Park [page 124].

Procedure

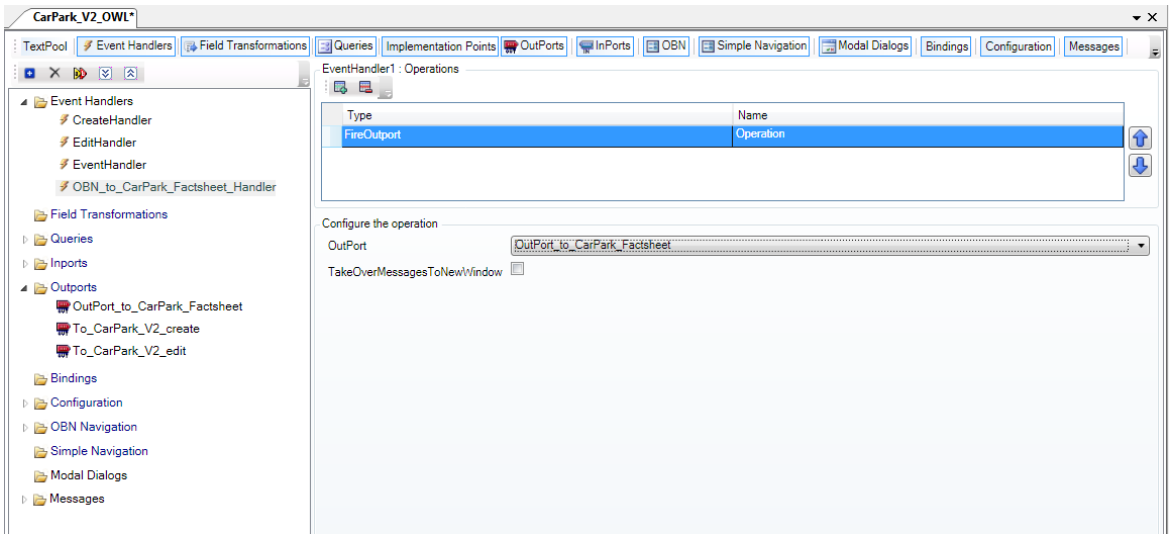
1. Open the Car Park OWL in the UI designer.
2. On the *Controller* tab, right-click *OBN Navigation* and select *Add OBN*. An outport is also created when you create the OBN.
3. Right-click the OBN and rename it to **`OBN_to_CarPark_fact sheet`**.
4. Check that the namespace and BO model are correct for your Car Park solution.
5. Click the CarPark node in the BO model and select *Open_CarPark_fact sheet* in the field *Select Operation*. The *PortType Package* and *PortType Reference* fields are populated automatically.



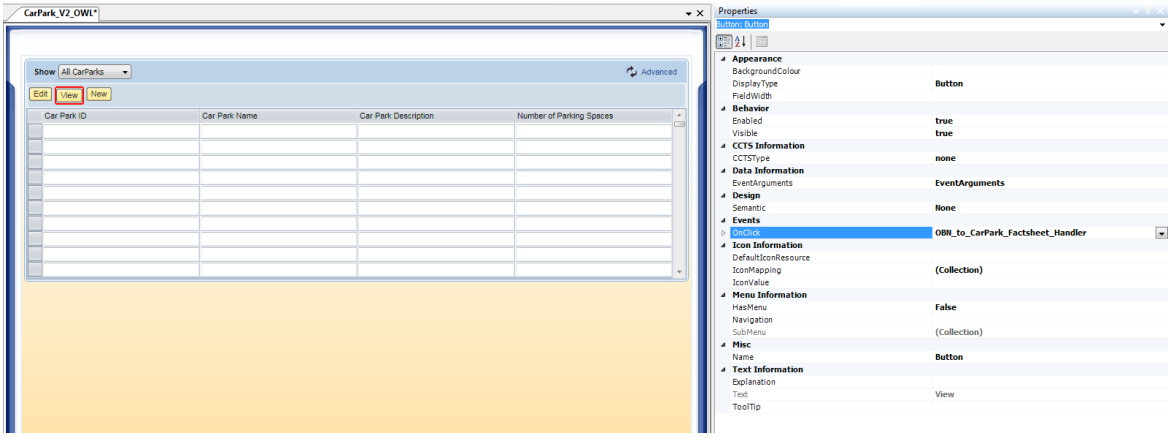
6. Right-click the outport that was created when you created the OBN and rename it to **`Outport_to_CarPark_fact sheet`**.
   You will notice that the *PortType Package* and *PortType Reference* fields are pre-populated.
7. Click the *Add Parameter* button.
8. In the parameter row, click the three dots next to *ParameterBinding*.
9. In the *Parameter Binding* dialog box, select the *NodeID* data field within the data list and click *OK*.



10. Right-click *EventHandlers* and select *Add Event Handler*.
11. Right-click the new handler and rename it to **`OBN_to_CarPark_fact sheet_Handler`**.
12. Click the *Add Operation* button.
13. Select the *Type* as *FireOutport* and select the *Outport* as *Outport_to_CarPark_fact sheet*.



14. Click the *Designer* tab and select the *ListPaneToolbar*.
15. In the *Property Explorer* select to configure the *Toolbar*.
16. In the *Configure Toolbar* dialog box, move the *View* button from *Available* to *Current* and click *OK*.
17. Select the *View* button and in the *Property Explorer* set the *Events > OnClick* field to *OBN_to_CarPark_fact sheet_Handler*.



18. Click *Save and Activate*.

5.8.3.6 Create an OBN to the Employee Fact Sheet

Overview

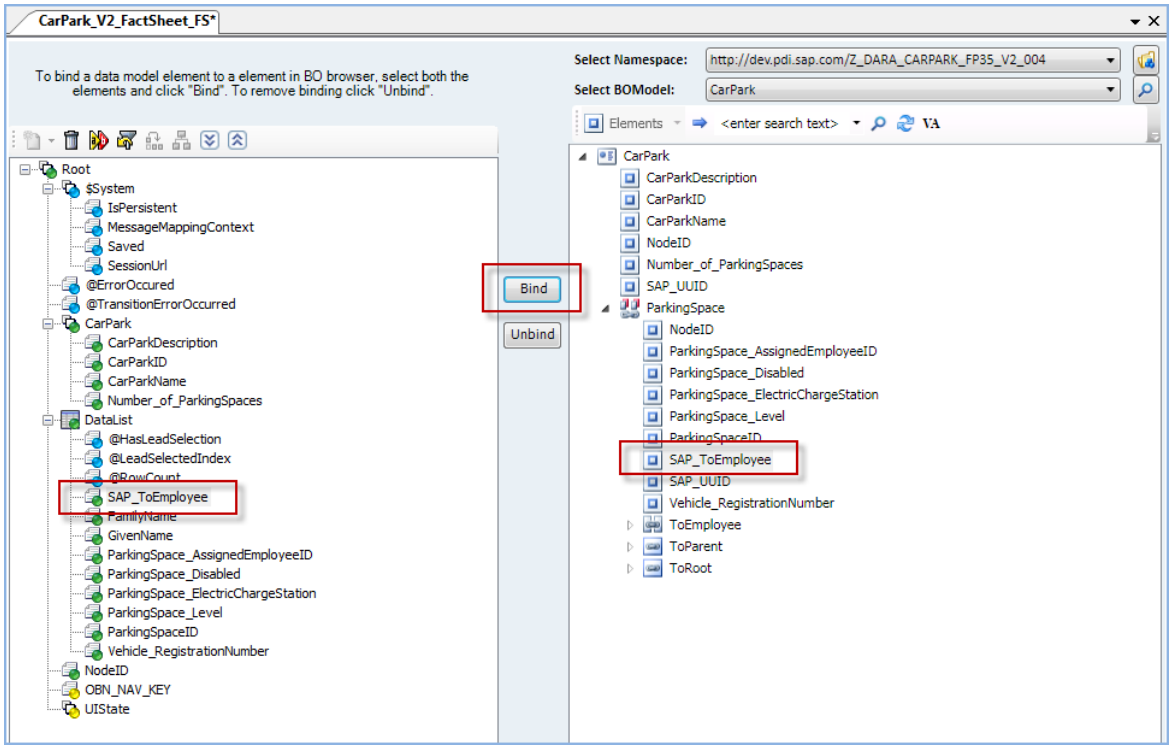
In this example you are going to create an object-based navigation (OBN) link from the Employee ID field on the Car Park fact sheet to the Employee's fact sheet.

Prerequisites

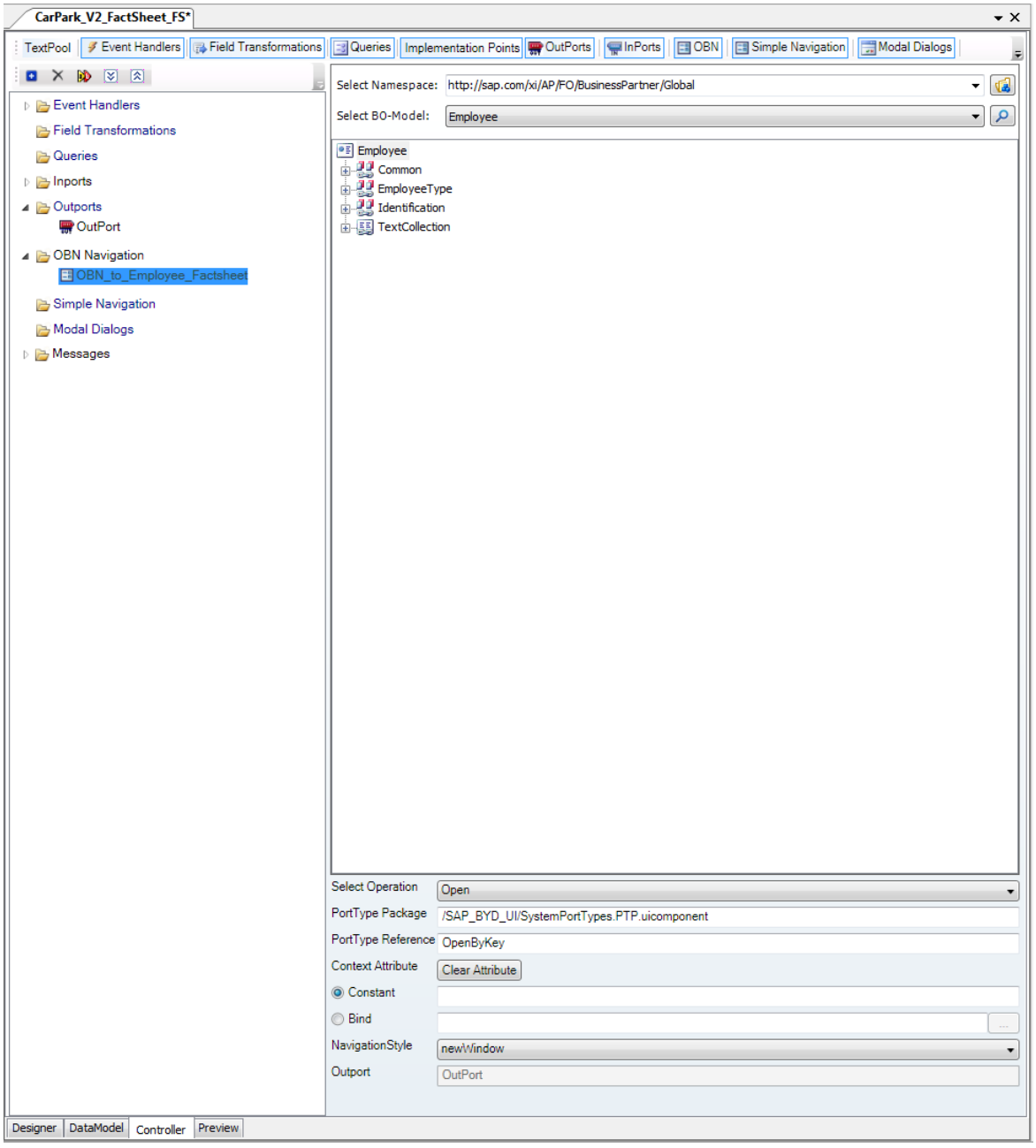
- You have completed the example: Building a Solution [page 99].
- You have completed the example: Create a Fact Sheet for the Car Park [page 124].
- You have completed the example: Create an OBN to the Car Park Fact Sheet [page 129].

Procedure

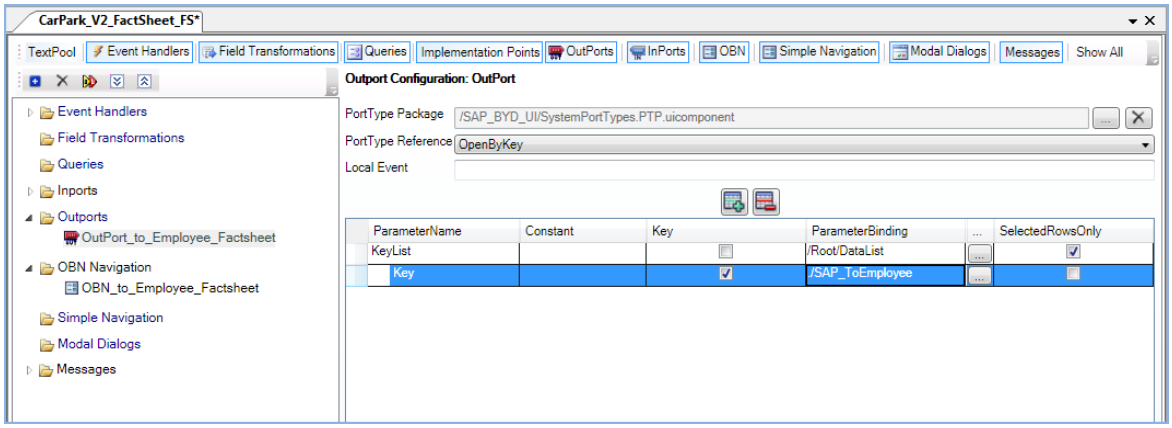
1. Open the Car Park fact sheet in the UI designer.
2. On the *DataModel* tab, add a new data field to the DataList.
3. Select the new data field, select the BO element *SAP_ToEmployee* and click the *Bind* button.



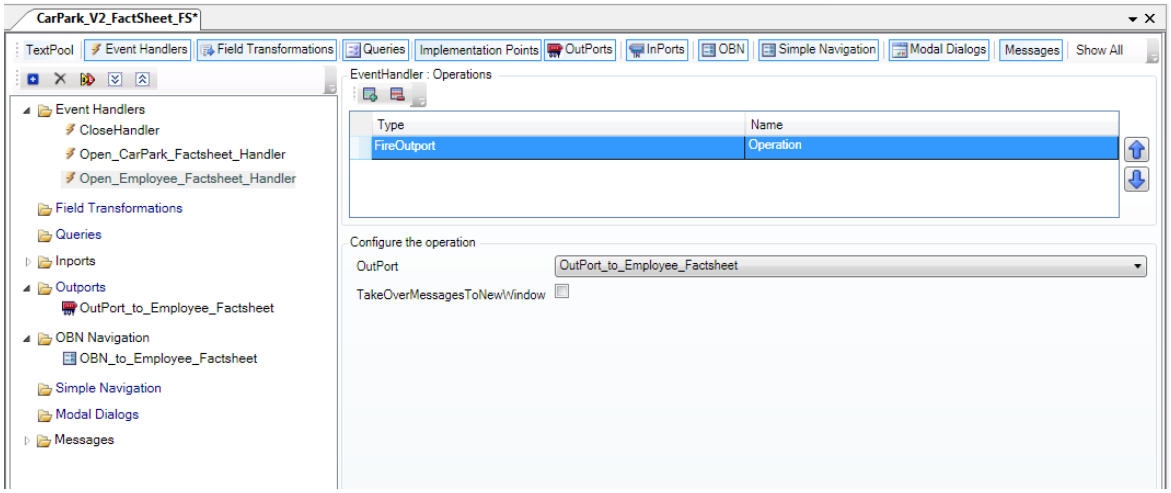
4. On the *Controller* tab, right-click *OBN Navigation* and select *Add OBN*. An outport is also created when you create the OBN.
5. Right-click the OBN and rename it to **`OBN_to_Employee_Factsheet`**.
6. Set the namespace to `http://sap.com/xi/AP/FO/BusinessPartner/Global`.
7. Select *Employee* for the BO model.
8. Select *Open* in the field *Select Operation*. The *PortType Package* and *PortType Reference* fields are populated automatically.



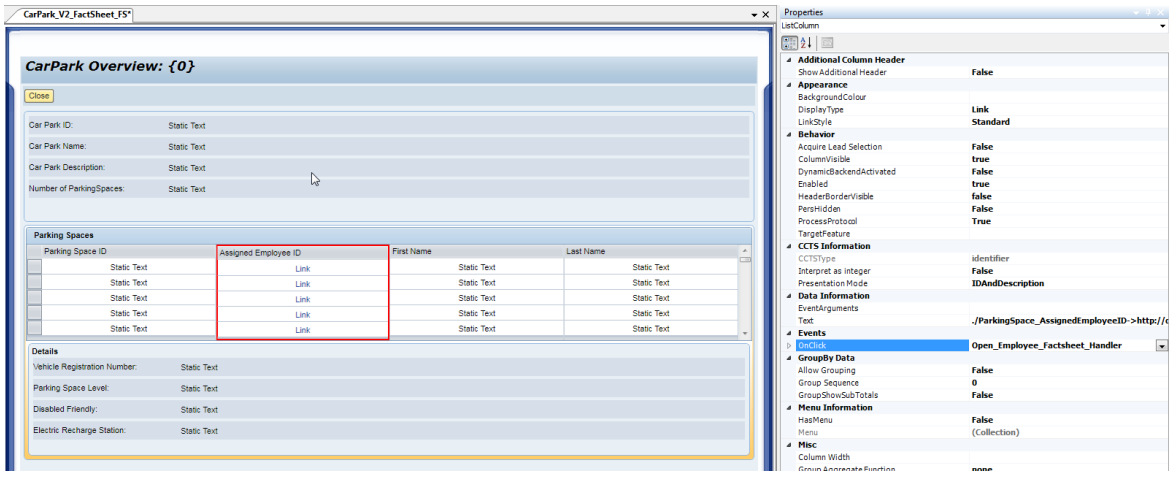
9. Right-click the outport that was created when you created the OBN and rename it to **`Outport_to_Employee_Factsheet`**.
   You will notice that the *PortType Package* and *PortType Reference* fields are pre-populated.
10. In the *KeyList* row, click the three dots next to *ParameterBinding*.
11. In the *Parameter Binding* dialog box, select the *DataList* and click *OK*.
12. In the *Key* row, click the three dots next to *ParameterBinding*.
13. In the *Parameter Binding* dialog box, select the *SAP_ToEmployee* data field within the data list and click *OK*.

14. 

15. Right-click *EventHandlers* and select *Add Event Handler*.
16. Right-click the new handler and rename it to **`Open_Employee_Factsheet_Handler`**.
17. Click the *Add Operation* button.
18. Select the *Type* as *FireOutport* and select the *Outport* as *Outport_to_Employee_Factsheet*.



19. Click the *Designer* tab and select the *Assigned Employee ID* column.
20. In the *Property Explorer* set the *Events > OnClick* field to *Open_Employee_Factsheet_Handler*.



21. Click *Save and Activate*.

5.8.3.7 Extend the Employee Business Object

Overview

In this exercise you will extend the Employee business object and then enhance the Employee fact sheet to display the car parking space assigned to the employee. This involves the following steps:

- create business object extension for the Employee business object
- add an extension field to the business object extension
- enhance the Employee fact sheet
- add the extension field to the fact sheet
- edit the AfterModify script to update the extension field

Prerequisites

- You have completed the example: Building a Solution [page 99]

Procedure

1. In the *Solution Explorer*, right-click your project and select *Add* > *New Item*.
2. Select the *Business Object Extension* template, enter the name **`Employee_BO_Extension`**, and click *Add*.
3. In the *Business Object Selection* dialog box, select the namespace `AP.FO.BusinessPartner.Global` and the business object `Employee`.
4. Open the business object extension and add the following code to the `Common` node:

```
node Common {

[Tooltip ("If the employee has been assigned a car parking space, the ID is
displayed here.")]
[Label ("Car Parking Space")]
element Assigned_Car_Parking_Space_ID : Text;

}
```

5. Save and activate the XBO.
6. In the *Solution Explorer*, right-click the *Employee_BO_Extension* and select *Enhance Screen*.
7. Select the floorplan `employeecontactdataqafeco` and click *OK*.
   The floorplan is opened in the UI designer.
8. On the *Designer* tab, select the *Basic Information* section group.
9. In the *Extensibility Explorer*, click the *Add Extension Field to Section Group* button.
10. In the *Add Extension Field* dialog box, select the extension field to add to the floorplan and click *Apply*.
    Remember to select the *Is Display Only* check box so that the field will appear as static text.
11. In the *Solution Explorer*, open the *Event-AfterModify* script for the Parking Space node.

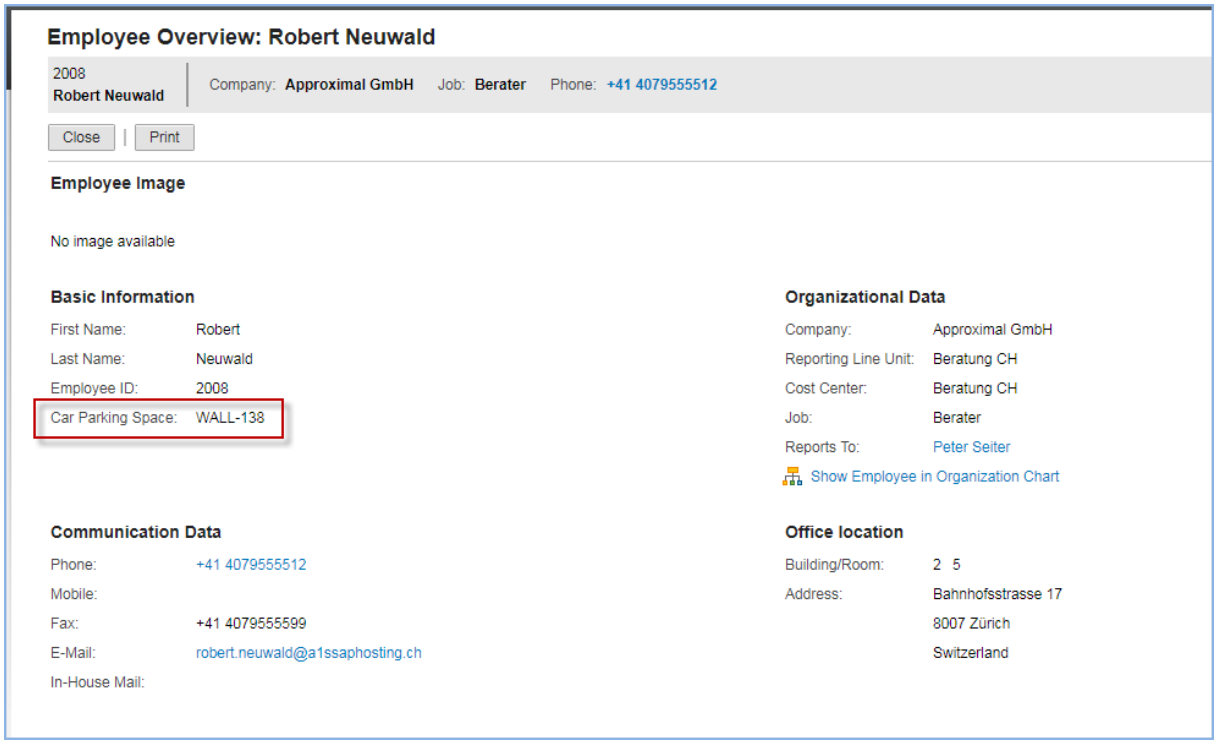
12. Add the following code at the end of the script file, just before the closing bracket:

```
foreach (var common in employee.Common)
{common.Assigned_Car_Parking_Space_ID = this.ParkingSpaceID;}
```

    When a car space is modified, this script will update the extension field (`Assigned_Car_Parking_Space_ID`) with the ID of the Parking Space.
13. Save and activate the script file.

Result

For employee's that have an assigned car parking space, the ID of the parking space is displayed on the employee's fact sheet.



> ### Note
>
> Because the value gets assigned in the AfterModify script for the Parking Space node, it will not affect existing entries. You need to create a new entry in the car park to see the result on the Employee fact sheet.

5.8.3.8 Validate On Save

Overview

In this example you are going to create a validation script that will check if there are any car spaces available. The script checks if the number of parking spaces assigned is greater than the total number of parking spaces. If there are no available spaces, an error message is displayed and you will not be able to save your changes to the car park.

Prerequisites

- You have completed the example: Building a Solution [page 99]

Procedure

1. Edit the car park business object to declare the message and specify the message text.
   1. Declare the message after the business object name:

      ```
      businessobject CarPark_Manager raises Message_Car_Park_Full {
      ```

   2. Enter the message text at the root node level:

      ```
      message Message_Car_Park_Full text "The car park is full - no spaces
      available";
      ```

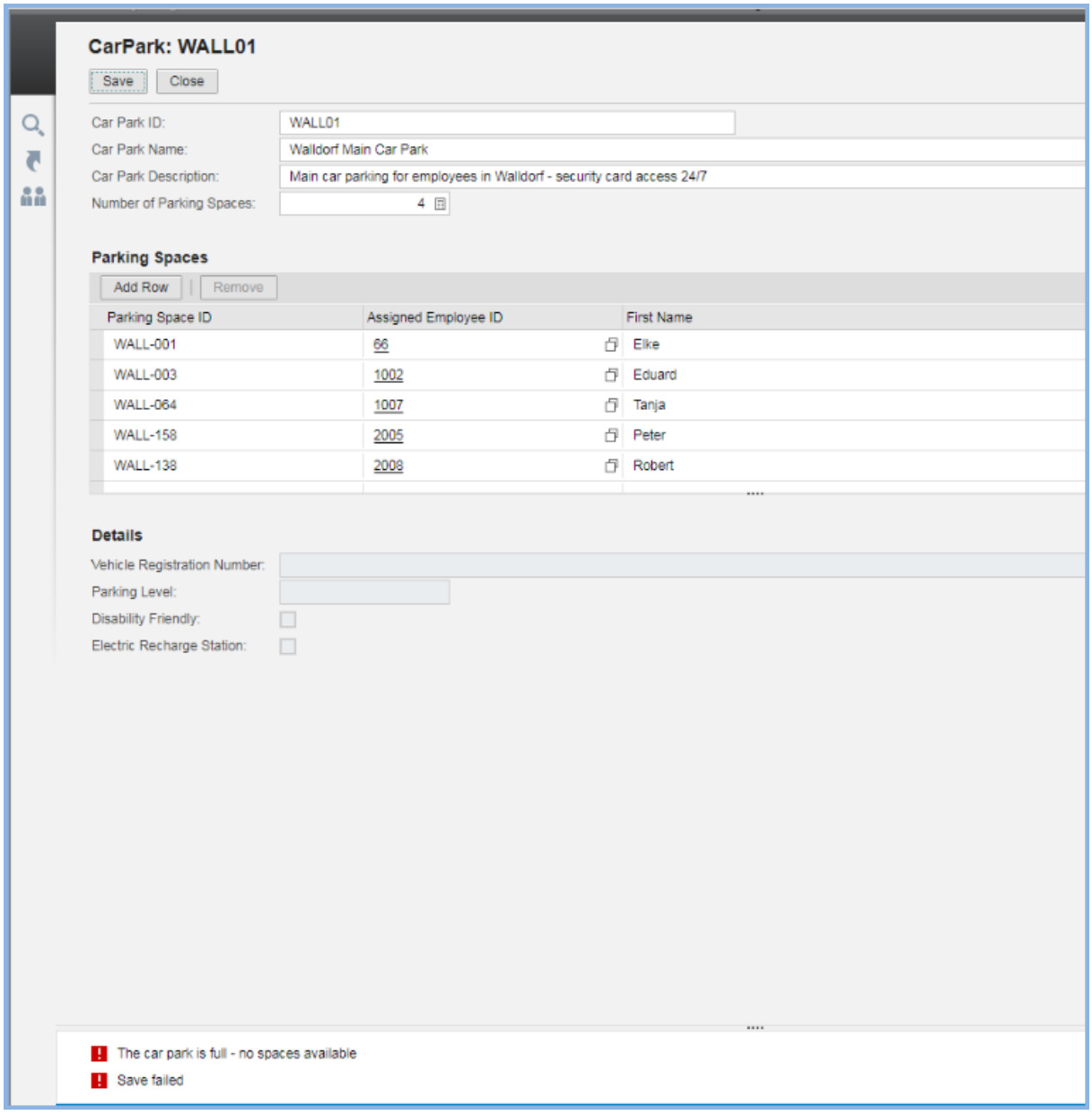
2. Save and activate.
3. Right-click the business object and select *Create Script Files*.
4. Create the *Validation: On Save* script file for the root node.
5. In the *Validation: On Save* script file, enter the following code:

   ```
   if (this.ParkingSpace.Count() > this.Number_of_ParkingSpaces)
   {
   raise Message_Car_Park_Full.Create("E");
   return false;
   }
   return true;
   ```

6. Save and activate.

Result

When you edit a car park and create more car spaces than the total number of car spaces, an error message is displayed and you will not be able to save your changes.



Error message

5.9 Using the Localization Toolkit (SAP Business ByDesign)

Overview

The localization toolkit is a set of development tools and instructions that enable SAP partners and customers to extend and adapt the capabilities of the SAP Business ByDesign solution to provide a locally-compliant solution. The localization toolkit enables customers and its subsidiaries to implement business processes for a number of localization topics supported by SAP Business ByDesign.

The localization toolkit leverages the SAP Business ByDesign business configuration, its extensibility feature (Key User Tool) and the Cloud Applications Studio (software development kit) to support the enhancements by

customers or partners. Using the localization toolkit, for example, you can implement additional business configuration, such as accounting principles and chart of accounts for a new country, create tax content for non-localized countries, create new reports, or implement different form templates for output documents.

The following topics are covered in the localization toolkit:

- General Ledger
- Tax
- Tax Reporting
- Fixed Assets
- Inventory
- Accounts Payable/Accounts Receivable
- Legal Reporting/Fiscal Closing
- Invoicing
- Payments
- Internationalization
- Languages
- Extensibility Features

Prerequisites

You have downloaded the SAP Cloud Applications Studio from the SAP Software Download Center at https://support.sap.com/swdc. For information about the installation steps and the latest builds, please see Installing and Updating the SAP Cloud Applications Studio [page 30].

Accessing the Toolkit

You can access detailed information about the localization toolkit topics at SAP Business ByDesign - Localization Toolkit. If you do not have access, please get in touch with your contact at SAP.

6 Administration and Lifecycle Management

6.1 Overview of Administration and Lifecycle Management

Administration and Lifecycle Management encompasses the following reference material:

Get Informed About Administration and Lifecycle Management of Customer-Specific Solutions

- Administration Quick Guide [page 159]
  The *Administration* toolbar provides access to a number of administrative tasks, such as creating a new solution, deleting a solution, and managing user sessions.
- Implementation Manager Quick Guide [page 162]
  You use the *Implementation Manager* to manage the lifecycle of customer-specific solutions.
- Lifecycle Management [page 143]
  The studio supports the full end-to-end lifecycle management of customer-specific solutions. This includes developing, testing, and deploying your solution, as well as creating patches to deliver updates to your customer.
- Developing on a Customer's Tenant [page 149]
  You can develop a customer-specific solution on the customer's test tenant. The finalized solution is then uploaded and activated on the customer's production tenant.
- Developing on Your Development Tenant
  - Lifecycle Management on Your Development Tenant [page 155]
    You can develop a customer-specific solution on your development tenant and after having tested, assembled, and downloaded the solution, you can upload it to the customer's tenant.
  - Switch a Customer Assignment [page 158]
    If you are developing customer-specific solutions on your development tenant, you can create solutions for more than one customer. For example, if you are working on a solution for customer A, you can save your solution and then switch to work on the solution for customer B. For this, you need to switch the customer assignment. Afterwards, in the My Solutions tool window in the studio, only the solutions of customer B are listed and you can create new solutions only for this specific customer.
- Quality Criteria [page 144]
  Customer-specific solutions for the SAP cloud solution are installed in a multi-tenant environment. The solution provider needs to ensure that all solutions are of the highest quality and have the lowest possible Total Cost of Ownership (TCO). SAP requests you to conform to the criteria described in this document in order to ensure the quality of your solution.

Get Informed About Solution Templates, Patches, Maintenance Mode, and Upgrade Information

- Solution Templates [page 171]
  With a solution template you can organize common development content that you want to reuse for customer-specific solutions. Solution templates enable you to rapidly start the development of customer-specific solutions, for example, for a specific industry
- Patches [page 178]
  A patch is needed if you want to make changes or corrections to a customer-specific solution after it has been assembled. The customer-specific solution is then in maintenance mode and you can only make restricted changes for certain content types.
- Maintenance Mode [page 185]
  A customer-specific solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is Assembled. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.
- Upgrade Information [page 17]
  Upgrade information is relevant if your customer's system is upgraded to a new release. The upgrade is maintained by SAP to ensure high quality upgrade of the customer's production tenant. The upgrade does not have any impact on your customer-specific solution. However, be aware of the downtime period during the upgrade and follow the rules listed in this document.

> **i** Note
>
> You can refer to SAP Cloud Applications Studio Deployment & Landscape Basics to better understand the basics of the landscape and the deployment of solution using **SAP Cloud Applications Studio**.

Availability

> **⚠** Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

6.2 Lifecycle Management

Overview

The studio supports the full end-to-end lifecycle management of customer-specific solutions. This includes developing, testing, and deploying your solution as well as creating patches to deliver updates to your customer

There are different scenarios in which you may want to develop a customer-specific solution:

- **Developing on a Customer's Test Tenant**
  You can develop a customer-specific solution on the customer's test tenant. The finalized solution is then uploaded and activated on the customer's production tenant. For more information, see Lifecycle Management on a Test Tenant [page 149].
- **Developing on Your Development Tenant**
  You develop your customer-specific solution on your development tenant in your company or organization. On this tenant you can develop solutions for different customers. The finalized solution is then downloaded from your development tenant and uploaded and activated on the customer's production tenant. For more information, see Lifecycle Management on Your Development Tenant [page 155].

> **i Note**
>
> The following guidelines and restrictions apply to the development of customer-specific solutions in general:
>
> - We recommend that you develop only one solution per customer.
> - The tenant where you develop your solution and the tenant where you upload your solution must be on the same release.

Tenant-Specific Activities in the Studio

The following table shows which activities you can perform on which tenant type.

Activity	Development Tenant	Customer's Test Tenant	For information, see ...
Create Solution	Yes	Yes	Create a Solution [page 177]
Create Copy of Solution	Yes	Yes	Implementation Manager Quick Guide [page 162]
Create Patch	Created in same namespace	Created in **separate** namespace	Patches [page 178]
Create Solution Template	Yes	Yes	Solution Templates Quick Guide [page 171]

Activity	Development Tenant	Customer's Test Tenant	For information, see ...
Create Copy of Template	Yes	Yes	Solution Templates Quick Guide [page 171]
Switch Customer	Yes	No	Switch a Customer Assignment [page 158]
Create and Maintain Solution With BC Sets Using SAP BCOs	Yes	No	Create a BC Set Using an SAP BCO [page 383]

6.3 Quality Criteria for Customer-Specific Solutions

Customer-specific solutions for the SAP cloud solution are installed in a multi-tenant environment. The solution provider needs to ensure that all solutions are of the highest quality and have the lowest possible Total Cost of Ownership (TCO).

SAP requests you to conform to the criteria described in this document in order to ensure the quality of your solution.

Functional Correctness

You must ensure that your solution has been tested and documentation is provided for all test cases.

Performance

You must ensure that your solution does not adversely impact the performance of the SAP cloud solution.

Response Time

Where the solution extends an existing user interface, a performance degradation of maximum 10% is acceptable even if the response time exceeds the target value.

Simple Transactions

- On average end-to-end < 1 s
- 95% end-to-end < 1.5 s

Examples of simple transactions are: Object Work Lists, purchase request, leave request.

Complex Transactions

The end-user response time for complex transactions must be:

- On average end-to-end < 1.7 s
- 95% end-to-end < 2.5 s

Examples of complex transactions are: integrated transactions like order, but also simple analytical content and fact sheets.

Complex Tasks

To enhance the end-user experience for complex tasks, the following must be true:

- Progress indicator available
- Background processing possible

Examples of complex tasks are: analytical content, planning, multilevel bill of material explosion.

Security

You must implement measures to ensure the security of your solution. Furthermore, your solution must not impact the security of the SAP cloud solution.

Code Scan

Security code scans are an important measure to minimize security risks. For customer-specific solutions, no code scans are performed by default. SAP reserves the right to perform code scans of customer-specific solutions in order to detect possible security risks.

If your solution functionality runs on an external system, you must conduct a security code scan or assign a third party to conduct a security code scan of the external component. SAP reserves the right to request proof of a successful code scan.

As there is no default security code scan for customer-specific solutions, you must take action to avoid the top 10 security risks as defined by the Open Web Application Security Project:

- http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Preventative Measures

You must implement measures to prevent security attacks. You can find more information on preventing common security attacks at the following links:

- Cross-site scripting attacks (http://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- SQL injection attacks (http://www.owasp.org/index.php/SQL_Injection)
- Directory traversal attacks (http://www.owasp.org/index.php/Path_Traversal)

> **Note**
>
> Useful guidelines and hints on secure programming can be found in the SAP NetWeaver Developer's Guide: http://help.sap.com/saphelp_nw2004s/helpdata/en/58/4d767ed850443c891ad27208789f56/frameset.htm

Backdoors

The solution must be free of intentionally hidden and non-documented features that circumvent security measures (backdoors).

Vulnerabilities

The solution must prevent typical vulnerabilities such as:

- Missing input validation
- Execution of commands that include not-validated user input
- Parameter tampering
- Cross-site request forgery
- Information leaks (for example, in error messages or hidden HTML fields)

> **Note**
>
> Information on Web-application security can be found here:
>
> - http://www.owasp.org/index.php/OWASP_Top_Ten_Project
> - http://cwe.mitre.org/top25/index.html

Documentation

Documentation must be provided to describe the following:

- Authorization and roles
- Communication security (including port information)

- User management and authentication
- Secure configuration of the service
- Accessing the security logs
- Configuring authentication policies (for example, behavior after a defined number of failed login attempts)
- Restricting access to specific IP addresses

You must provide customers with updates to the security documentation (for example, security bulletins) when applicable.

Authorizations

The solution must provide the capability to separate the authorizations (roles) for different types of users (that is, administrative and non-administrative users). Users must only have the permissions that are required for their work (segregation of duties). The authorization concept must be documented (that is, delivered roles per business and technical user). Search results may contain sensitive data and therefore must be filtered based on the authorization level of the user.

Data Transfer

The transfer of data to other services or service providers must be configurable and documented.

Testing

Details must be provided of any security-related test activities performed.

Security Settings

Security settings and recommendations must be documented and available to the customer.

Supportability

You must provide for the maintenance and support of your partner solution throughout its entire product lifecycle.

Maintenance and Support – Customer

Customer Service Level Agreement (SLA)

As you are fully responsible for the solution, you have a support contract with your customer. You must describe any Customer SLA or Customer Support Agreement that exists for your partner solution.

Maintenance

Your solution must have a defined release and maintenance strategy and process, which must be linked with the SAP interface, product, and maintenance process for the SAP cloud solution. This is required as SAP reserves the right to deprecate entities or functions in the SAP cloud solution with a new release, which might have an impact on your solution and might require that you need to update your solution.

Maintenance and Support – SAP

Support Process

Support processes/interaction between your support department and SAP Support must be aligned and documented.

Interface Deprecation

You must be able to react, properly and in time, to the possible deprecation of interfaces from SAP's public solution model (PSM).

Interfaces of the PSM can be deprecated by SAP. Generally, SAP provides alternative interfaces that continue to offer the required business functionality.

SAP promises that deprecated interfaces remain fully functional within two releases (approximately one year). During this period of time, you need to react by removing usage of the deprecated interfaces from your partner solution.

Public Solution Model Access

You must only access SAP interfaces according to the PSM.

Switching the Solution On/Off

The SAP cloud solution must remain fully functional if a customer-specific solution is switched off. If your solution jeopardizes the standard functionality of the SAP cloud solution, SAP reserves the right to switch off your solution.

If your solution is switched on again, it then must be able to resynchronize to the existing data in the system.

6.4 Developing on a Customer's Tenant

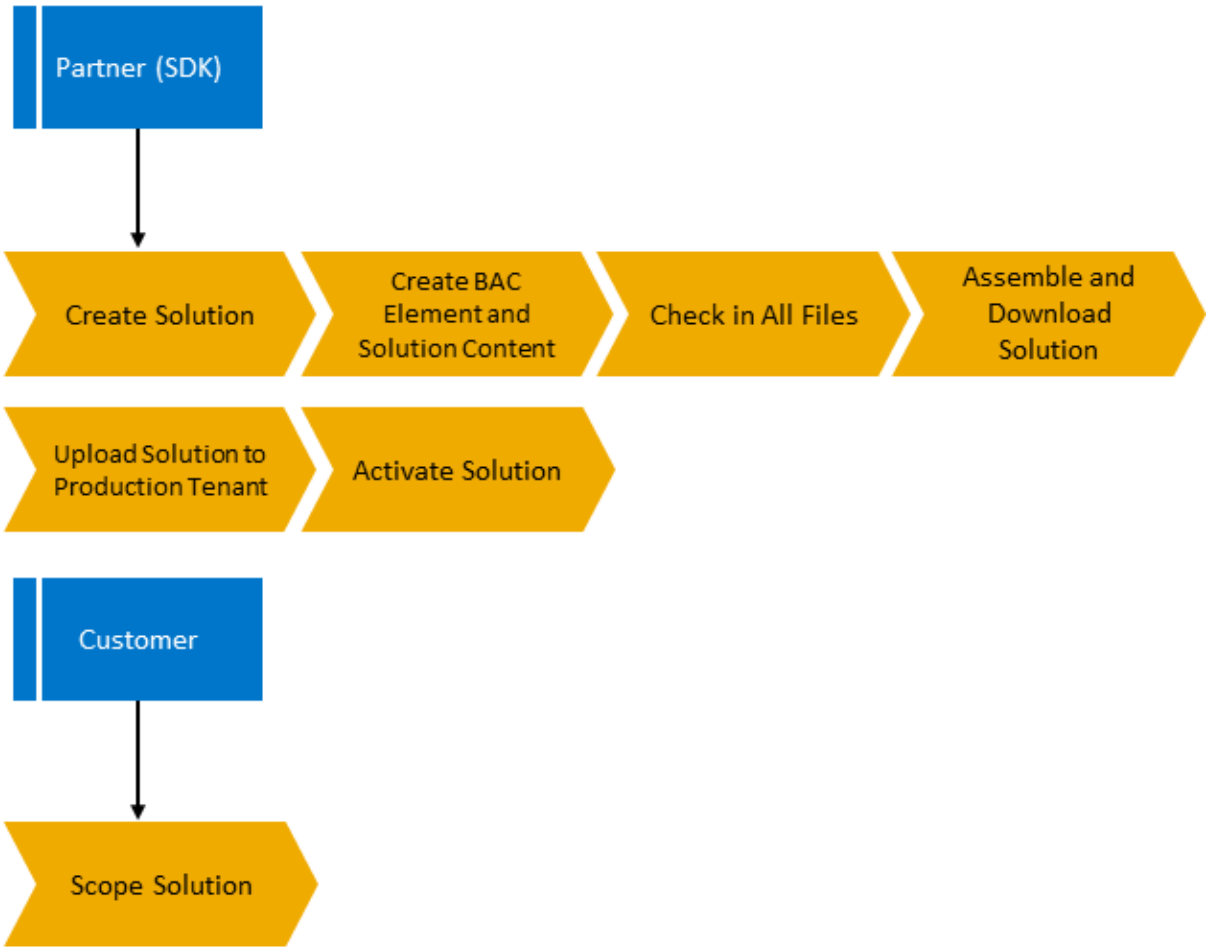
6.4.1 Lifecycle Management for Customer-Specific Solutions on a Customer's Test Tenant

Overview

If the customer has already gone live with the SAP solution, you can develop and test your customer-specific solution on the customer's test tenant. In this case, development is done on the test tenant and the finalized solution is uploaded and activated on the customer's production tenant. This document describes the after go live scenario. For information about how to request a test system, see the documentation of the SAP Solution.

> **i** Note
>
> The following guideline applies if you work on a test tenant:
>
> - If the test tenant is a copy of the customer's production tenant, on which your solution is already activated, you can only create a patch for this solution. For more information, see Create a Patch on a Customer's Test Tenant [page 152].

The following graphic gives a schematic overview of the lifecycle management process. The process is explained in detail below.



Prerequisites

In the SAP cloud solution, your customer has set up the following users for you:

- A business user for testing purposes in the SAP cloud solution on the test tenant.
- Users with the roles **Studio Administrator** and **Developer** in any tenants in which you are going to develop and deploy the solution. If you both develop and implement the solution, you only need a user with the **Studio Administrator** role. For more information, see User Setup Quick Guide [page 33] and User Roles [page 32].

Process Flow

1. In the studio, you log on to the customer's test tenant and create a solution (**Studio Administrator**).
2. You develop your solution including the necessary business configuration content (**Developer**).

> **i Note**
>
> You can create BC Sets using SAP BCOs only on your development tenant.

3. You test the solution (**Business User**).

   You can test specific solution content in the studio using the preview function.

   You can also test your solution in the SAP cloud solution for the tenant on which you created the solution. As a prerequisite, you must scope your solution. You have the following options:

   - In the studio, you trigger deployment of your business configuration content (**Studio Administrator**).

     > **i Note**
     >
     > If you have defined BAC elements for your solution, a dialog box opens when you use the *Deploy Business Configuration* function. You have to select one of the following options:
     >
     > - Deploy all business configuration content for your solution
     > - Deploy only the business configuration content for your solution that is assigned to the business options you have selected in the business adaptation catalog
     >
     > This allows you to test both partial and complete activation of your business configuration content in the development environment.

     You then log on to the SAP cloud solution with your business user and perform your tests.
   - In the SAP cloud solution, you scope your solution manually (**Business User**).
     To do this, you log on to the SAP cloud solution with your business user. In the *Business Configuration* work center, you perform scoping for the business option that you created for your solution and for the required standard functions of the SAP cloud solution.
     You then perform your tests.
4. You assemble and download the solution (**Studio Administrator**).
   After you have performed your tests successfully, in the studio, you log on to the test tenant to assemble and download your solution. For more information, see the Implementation Manager Quick Guide [page 162].

   > **i Note**
   >
   > If you are working on a preproduction tenant and you have used the *Deploy Business Configuration* function in the previous step, the system now removes the business configuration content from the SAP cloud solution.

5. You upload the solution to the production tenant (**Studio Administrator**).

   > **i Note**
   >
   > If you are developing your solution on the customer's preproduction tenant, then you do not need to upload the solution. Instead, after assembling the solution, go to step 7.

   In the studio, you log on to the production tenant and upload the solution. .
6. You activate the solution on the production tenant (**Studio Administrator**).
   After the solution has been successfully uploaded, you activate it on the customer's production tenant.

   > **i Note**
   >
   > Before activating the solution, we recommend that you inform the customer of any functions that may be affected by the update. For example, if the administrator at the customer has adapted an analytical report, this may be affected by any changes you have made to the same report.

7. The customer activates your solution by selecting the relevant business option in Scoping.

Follow-On Activities

After the solution has been activated on the customer's production tenant, you can only make changes or corrections to it by creating a patch. A patch follows the same lifecycle as the original solution, with the exception that the business option does not need to be selected again.

6.4.2 Create a Patch on a Customer's Test Tenant

Overview

If you need to make changes or corrections to a customer-specific solution after you have assembled it, you can create a patch.

For more information, see Patches [page 178].

> **i** Note
>
> The following guidelines and restrictions apply:
>
> - Any data created for the original solution in the SAP cloud solution and any scoping decisions and fine-tuning settings are not copied to the patch solution. For testing purposes, you must therefore scope the patch solution and create new test data.
> - If your original solution contains a Web service, this Web service will have a different namespace in the patch solution. You must therefore adapt the settings of your client testing tool to test the Web service in your patch solution.

Prerequisites

In the studio, you have logged on to the customer's test tenant where you created the original solution and you have opened this solution.

> **i** Note
>
> Only users with the **Studio Administrator** role can create patches.

Procedure

1. On the *Implementation Manager* toolbar, click the *Create Patch* button.

The system creates a patch solution and copies all files in your solution to a new namespace within the patch solution. The namespace is generated automatically and cannot be changed.
The patch solution is created in the background and this process may take some time. Click *Refresh* in the *Implementation Manager* tool window to update the solution status.

2. In the *Implementation Manager*, open the patch solution by clicking the *Open* button next to the *Patch Solution* field.

> **→ Tip**
>
> You can always access the patch solution from the original solution by opening the *Implementation Manager* and, on the *Current Version* tab, clicking the *Open* button next to the *Patch Solution* field. You can also navigate to the original solution from the patch solution in the same way.

3. In the patch solution, make your changes. Then activate the patch solution.
4. You can test specific solution content in the studio using the preview function. To test your patch solution in the SAP cloud solution, you have to enable the solution on the test tenant as follows:
    1. In the *Implementation Manager*, click the *Enable* button next to the *Solution Enablement for Business User in Cloud Solution* field.
    The "Disabled" changes to an "Enabled" and the patch solution is enabled on the test tenant instead of the original solution. For more information, see Enable and Disable Your Solution in the SAP Cloud Solution [page 182].

    > **i Note**
    >
    > If a BC change project is open on the tenant before you enable the patch solution on the test tenant, you will not be able to merge the change project unless you disable the patch solution. For more information, see What You Need to Consider When a Change Project Is on the Same Test Tenant [page 154].

    2. In the SAP cloud solution, log on to the test tenant with your business user.
    You will be able to see your patch solution in place of the original solution.
    3. Complete your testing.
5. On the *Implementation Manager* toolbar, click the *Assemble and Download* button.
The system assembles the patch using the namespace of the original solution and downloads it to a `.zip` file. This allows you to copy your corrections back to the original solution.
6. In the studio, log on to the production tenant on which the original solution is active, and click the *Upload* button on the *Implementation Manager* toolbar. Select the `.zip` file containing the patch.
The system uploads the `.zip` file with your corrections and checks for any compatibility issues. The solution has the status *In Deployment* and the solution version number is updated.
7. In the *Implementation Manager*, select the solution and click *Activate*.

> **i Note**
>
> Before activating the solution, we recommend that you inform the customer of any functions that may be affected by the update.

6.4.3 What You Need to Consider When a Change Project Is on the Same Test Tenant

If you are developing customer-specific solutions on your customer's test tenant and your customer uses this tenant for change projects at the same time, both of you need to consider the following information. Otherwise your customer won't be able to merge the change project to the production tenant.

This information is not relevant when your customer has requested a separate test tenant that is only used for the change project.

> **i** Note
>
> As a general rule, the solutions on both tenants need to have the same version when the change project is merged to the customer's production tenant. Another rule is that a change project cannot be merged if a solution template is being developed on the customer's test tenant or has been uploaded to it. In such a case, you must disable the solution template on the test tenant **before** the change project is merged.

Check this table to find out what needs to be done before the change project can be merged to the production tenant. Answer the following questions and make a decision:

- Did a solution exist on the production tenant **before** the change project was created?
- Do you intend to create a solution or patch solution on the test tenant **after** the change project has been created?

Did a solution exist on the PRODUCTION tenant?	Will you create a solution on the TEST tenant?	What do you need to do?
No, there was no solution.	Yes, I'll create a solution.	Before the change project can be merged to the production tenant, you need to do *one* of the following: • Complete your solution, assemble and download it, and upload the solution to the production tenant. • If you don't want to upload the solution to the production tenant, disable it for the business user in the test tenant. For more information, see Enable or Disable Your Solution in the SAP Cloud Solution [page 182].
Yes, a solution existed.	Yes, I'll create a patch solution. I'll also enable it for the business user to allow the patch to be tested.	Before the change project can be merged to the production tenant, you need to do the following: • Disable the patch solution for the business user in the test tenant. • Make sure that the original solution is enabled. For more information, see Enable or Disable Your Solution in the SAP Cloud Solution [page 182].

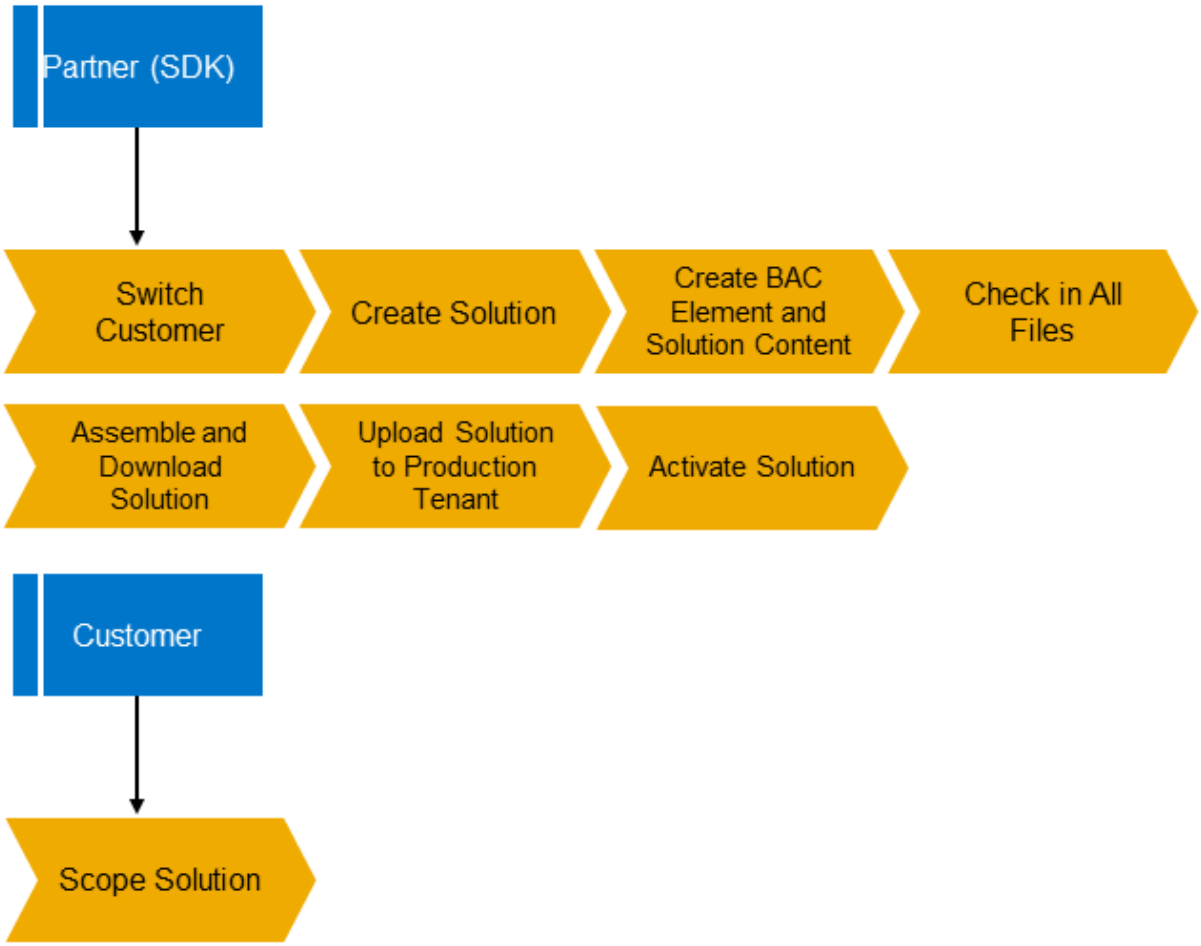
6.5 Developing on Your Development Tenant

6.5.1 Lifecycle Management of Customer-Specific Solutions on Your Development Tenant

Overview

You can develop a customer-specific solution on your development tenant and after having tested, assembled, and downloaded the solution, you can upload it to the customer's tenant.

The following graphic gives a schematic overview of the lifecycle management process. The process is explained in detail below.



Prerequisites

- In the SAP cloud solution, you have set up the following users:
  - Users with the roles **Studio Administrator** and **Developer** on the tenant on which you want to develop and assemble the solution. For more information, see User Setup Quick Guide [page 33] and User Roles [page 32].

- User with the **Studio Administrator** role on any tenants to which you want to upload the solution, for example, the customer's production tenant.
- A **business user** for testing purposes in the SAP cloud solution on your development tenant.

- In the studio, you are logged on to your development tenant.

Process Flow

1. In the studio, you check the current customer assignment and make sure that the ID of the customer for whom you want to create and develop the solution is displayed (**Studio Administrator**).
   For more information, see Switch a Customer Assignment [page 158].
2. You create a solution (**Studio Administrator**).
3. You develop your solution including the necessary business configuration content (**Developer**).
4. You test the solution (**Business User**).

   You can test specific solution content in the studio using the preview function.

   You can also test your solution in the SAP cloud solution for the tenant on which you created the solution. As a prerequisite, you must scope your solution. You have the following options:

   - In the studio, you trigger deployment of your business configuration content (**Studio Administrator**).

     > **i Note**
     >
     > If you have defined BAC elements for your solution, a dialog box opens when you use the *Deploy Business Configuration* function. You have to select one of the following options:
     >
     > - Deploy all business configuration content for your solution
     > - Deploy only the business configuration content for your solution that is assigned to the business options you have selected in the business adaptation catalog
     >
     > This allows you to test both partial and complete activation of your business configuration content in the development environment.

     You then log on to the SAP cloud solution with your business user and perform your tests.
   - In the SAP cloud solution, you scope your solution manually (**Business User**).
     To do this, you log on to the SAP cloud solution with your business user. In the *Business Configuration* work center, you perform scoping for the business option that you created for your solution and for the required standard functions of the SAP cloud solution.
     You then perform your tests.
5. You assemble and download the solution (**Studio Administrator**).
   For more information, see the Implementation Manager Quick Guide [page 162].
6. After you have performed a tests successfully, you log on to the customer's production tenant in the studio to upload the solution. You activate the solution on the production tenant (**Studio Administrator**).

   > **i Note**
   >
   > Before activating the solution, we recommend that you inform the customer of any functions that may be affected by the update. For example, if the administrator at the customer has adapted an analytical report, this may be affected by any changes you have made to the same report.
7. In the SAP cloud solution, the customer activates a solution by selecting the relevant business option in scoping.

Follow-On Activities

After a solution has been activated on the customer's tenant, you can only make changes or corrections to it by creating a patch. A patch follows the same lifecycle as the original solution, with the exception that the business option does not need to be selected again. For more information, see Create a Patch on Your Development Tenant [page 157].

6.5.2 Create a Patch on Your Development Tenant

Overview

If you need to make changes or corrections to a customer-specific solution after you have assembled it, you can create a patch.

When you create a patch of a solution on your development tenant, the patch is created in the same solution and namespace but with a different version number. For more information, see Patches [page 178].

Prerequisites

In the studio, you have logged on to your development tenant where you created the original solution and you have opened this solution.

> **i** Note
>
> Only users with the **Studio Administrator** role can create patches.

Procedure

1. On the *Implementation Manager* toolbar, click the *Create Patch* button.
   The system creates a patch in the same solution and sets the solution status to *In Development*. The version number of the solution is updated to the next higher number.
2. Your can now implement the required changes.
   When you have finished making your changes, you can activate and then test your solution.
3. In the *Solution Explorer*, right-click your solution and choose *Activate*.
4. To test your solution, you can do one of the following:
   - In the studio, trigger deployment of your business configuration content. To do this, in the *Solution Explorer*, right-click the *Business Configuration* node in your solution and select *Deploy Business Configuration*.

> **i Note**
>
> If you have defined BAC elements for your solution, a dialog box opens when you use the *Deploy Business Configuration* function. You have to select one of the following options:
>
> - Deploy all business configuration content for your solution
> - Deploy only the business configuration content for your solution that is assigned to the business options you have selected in the business adaptation catalog
>
> This allows you to test both partial and complete activation of your business configuration content in the development environment.

You then you log on to the SAP cloud solution with your business user and perform your tests.

- In the SAP cloud solution, you scope your solution manually.
  To do this, you log on to the SAP cloud solution with your business user. In the *Business Configuration* work center, you perform scoping for the business option you created for your solution and for the required standard functions of the SAP cloud solution.
  You then perform your tests.

When you have finished testing, you can download your solution and then upload it to the customer's tenant.

5. In the studio, in the *Solution Explorer*, right-click the project of your solution and select *Check In All Files*.
6. On the *Implementation Manager* toolbar, click the *Assemble and Download* button.
   The system assembles the patch and downloads it to a `.zip` file.
7. Log off from your development tenant and then log on to the customer's tenant to which you have uploaded the patch.
8. In the *Implementation Manager*, click the *Upload* button. Select the `.zip` file containing the patch.
   The system uploads the `.zip` file with your corrections and checks for any compatibility issues.
9. In the *Implementation Manager*, click the *Activate* button.
   The patch is activated in the cloud solution from SAP.

6.5.3 Switch a Customer Assignment

Overview

If you are developing customer-specific solutions on your development tenant, you can create solutions for more than one customer. For example, if you are working on a solution for customer A, you can save your solution and then switch to work on the solution for customer B. For this, you need to switch the customer assignment. Afterwards, in the *My Solutions* tool window in the studio, only the solutions of customer B are listed and you can create new solutions only for this specific customer.

Prerequisites

In the studio, you are logged on to your development tenant.

> **i** Note
>
> If your are logged on to your development tenant with a user that has the **Developer** role, you can only switch a customer assignment. However, if you want to create a new customer assignment, you need to log on with a user that has the **Studio Administrator** role.

Procedure

1. Choose *Administration* > *Switch Customer*.
2. In the *Switch Customer* dialog box, check the details of the current customer assignment. Each solution that you create, will be assigned to this customer. If you want to switch the assignment, you can do one of the following:
   - To switch to an existing customer assignment, select a customer from the list of existing customers.
   - To create a new customer assignment, click *Create Customer*.
     1. In the *Create Customer* dialog box, enter a customer name and the customer ID that SAP has provided to the customer.
     2. Click *OK*.
     3. In the *Switch Customer* dialog box, choose this new customer from the list.
3. In the *Switch Customer* dialog box, click *OK*.
   The system logs off and you need to log on again to your development tenant. In the *My Solutions* window, only the solutions of the current customer are listed. If you have not yet created any solutions for this customer, the *My Solutions* window is empty.

6.6 Administration

6.6.1 Administration Quick Guide

Users with the **Studio Administrator** role can use the *Administration* toolbar to access a number of administrative tasks, such as creating and deleting customer-specific solutions or solution templates and managing user sessions.

> → Tip
>
> If the *Administration* toolbar is not visible in the studio, you can access it by clicking *View* > *Toolbars* > *Administration*.

You can also use the functions for the studio administrator in the *Administration* menu.

Business and Technical Background

Lifecycle Management of Customer-Specific Solutions

The studio supports the full end-to-end lifecycle management of customer-specific solutions. This includes developing, testing, and deploying your solution as well as creating patches to deliver updates to your customer

For more information, see Lifecycle Management of Customer-Specific Solutions [page 143].

Solution Templates

You can create templates containing reusable items and upload these items to customer-specific solutions.

For more information, see Solution Templates [page 174] and Solution Templates Quick Guide [page 171].

Patches

If you need to make changes or corrections to a customer-specific solution after you have assembled it, you can create a patch.

For more information, see Patches [page 178].

Maintenance Mode

A customer-specific solution is in maintenance mode, if it has been assembled and downloaded; the solution status then is *Assembled*. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.

For more information, see Maintenance Mode [page 185].

Tasks

Create a Solution

You can design and develop customer-specific solutions to enhance the SAP standard cloud solution for a specific customer. In the studio, you can create a solution in which all items that you develop or changes that you make to existing entities are stored.

For more information, see Create a Customer-Specific Solution [page 177].

Create a Solution Template

You can create templates containing reusable items and upload these items to customer-specific solutions.

For more information, see here [page 175].

Create a Patch

If you need to make changes or corrections to a customer-specific solution after you have assembled it, you can create a patch. Depending on the tenant on which you are working, the process of creating a patch is different:

- Create a Patch on a Customer's Tenant [page 152]
- Create a Patch on Your Development Tenant [page 157]

Delete Locks Using the Session Administration

Locks occur when an item, such as a business object, is being edited by another user or from another session. As an administrator, you can delete locks for other users; all other users can only delete their own sessions and corresponding locks.

> **i** Note
>
> - Deletion of locks may result in the loss of changes made by the locking user, and should only be carried out when absolutely necessary.
> - Locks created by the UI designer can only be deleted by the user that created the lock; an administrator cannot delete these locks.

1. On the *Administration* toolbar, click *Session Administration*.
   The *Session Administration* opens.
2. Click the *Refresh* button to get a list of users with sessions.
3. Select the user for whom the session should be deleted.
4. Click the *Delete* button.
5. Make sure that the user whose session was deleted logs off the system. If the user continues working without logging off, locks without session will be created. These locks can only be deleted by the user who created the locks. If you are deleting your own session, you must also log off immediately afterwards to avoid creating locks without session.

Delete a Solution

1. Log on to the repository and open the solution.
2. On the *Administration* toolbar, click *Delete Solution* and confirm that you want to delete the solution.

> **i** Note
>
> You cannot delete a solution on a production tenant. We recommend you switch the solution off by deselecting it in Scoping. Then report an incident to request assistance from SAP.

Further Tasks

You use the *Implementation Manager* to manage the lifecycle of customer-specific solutions. The *Implementation Manager* enables you, for example, to assemble and download a solution, upload and activate a solution, and create a patch for a solution. If you are not able to activate a solution, you can carry out different troubleshooting activities in the *Implementation Manager*.

For more information, see Implementation Manager Quick Guide [page 162].

6.6.2 Implementation Manager Quick Guide

You use the *Implementation Manager* to manage the lifecycle of customer-specific solutions. The *Implementation Manager* enables you, for example, to assemble and download a solution, upload and activate a solution, and create a patch for a solution. If you are not able to activate a solution, you can carry out different troubleshooting activities in the *Implementation Manager*.

> **i** Note
>
> Only users with the Administrator role can access the *Implementation Manager* tools. For more information see User Roles [page 32].

If the *Implementation Manager* is not visible in the studio, you can access it as follows:

- To enable the *Implementation Manager* tool window, click *View* > *Implementation Manager*. The tool window provides a view of the solution status and implementation log at each stage in the process.
- To enable the *Implementation Manager* toolbar, click *View* > *Toolbars* > *Implementation Manager*. The toolbar provides buttons that allow you to assemble, upload, and create a patch for your solution.

Business and Technical Background

Lifecycle Management

The studio supports the full end-to-end lifecycle management of customer-specific solutions. This includes developing, testing, and deploying your solution as well as creating patches to deliver updates to your customer

For more information, see Lifecycle Management of Customer-Specific Solutions [page 143].

Patches

If you need to make changes or corrections to a customer-specific solution after you have assembled it, you can create a patch.

For more information, see Patches [page 178].

Maintenance Mode

A customer-specific solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is `Assembled`. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.

For more information, see Maintenance Mode [page 185].

Solution Templates

You can create templates containing reusable items and upload these items to customer-specific solutions.

For more information, see Solution Templates [page 174] and Solution Templates Quick Guide [page 171].

Tasks

Activate, Assemble, and Download a Solution

When you have finished developing your solution, you activate, assemble, and download it before uploading it to the customer's tenant. You can do this in 2 ways as shown below:

1. Assemble and download as a front-end process
    1. In the studio, log on to the tenant where you created the solution, and open the solution.
    2. In the *Solution Explorer*, right-click the solution and select *Activate*.
    3. In the *Solution Explorer*, right-click the project and select *Check In All Files*.
    4. On the *Implementation Manager* toolbar, click the *Assemble and Download* button.
    5. Enter a file name to save the template as a .zip file locally.

    The system checks the solution for consistency and then assembles and downloads all the items in the solution to a .zip file.
    After the solution has been assembled and downloaded, the status changes to *Assembled*. The solution is now locked for further development and any changes can only be made by creating a patch.
2. Split activate, and assemble as a background job
    1. In the studio, log on to the tenant where you created the solution, and open the solution.
    2. On the *Implementation Manager* toolbar, click the *Assemble and Download* button. You will find a dropdown option to *Enable Assembly Split*. This enables you to view the individually listed out *Activate*, *Assemble*, and *Download* actions .

    > **i Note**
    >
    > The activate and assemble functions run as background jobs, and you can continue to work on other solutions in the system. The download activity, however, is a front-end process.

    > **i Note**
    >
    > Depending on the status of the solution, only the relevant action will be available for use. The following actions will be activated and available for use for:
    >
    > - *Activate*: All solutions with **In Development** status
    > - *Assemble*: All solutions that have been successfully activated
    > - *Download*: All solutions that have been successfully assembled

    3. Click *Activate* to activate the solution. The system activates the solution and makes it ready for Assembly. Once completed, the system sends you an auto-generated e-mail stating the success or failure of the activation. You can view the details for failed actions by logging on to the studio, and selecting the *Version History* tab in the *Implementation Manager*.
    4. Click *Assembly*, and the system checks the solution for consistency and then assembles it. Again, you will receive an auto-generated e-mail with the details of the success or failure of the assembly process. You can view the details for failed actions by logging on to the studio, and selecting the *Version History* tab in the *Implementation Manager*.
    5. After the assembly is completed, click *Download* to download the zip file. Enter a file name to save the template as a .zip file locally.

    > **i Note**
    >
    > You can can download previous versions of the solution too.

After the solution has been assembled and downloaded, the status changes to *Assembled*. The solution is now locked for further development and any changes can only be made by creating a patch.

Upload and Activate a Solution

After you have assembled and downloaded your solution, you can upload it to the customer's tenant.

1. Log on to the tenant.
2. On the *Implementation Manager* toolbar, click the *Upload* button and select the `.zip` file created in the assemble and download step.
   An upload disclaimer appears.
3. Agree to the terms of the upload disclaimer and click *Continue*.
   The system unpacks the `.zip` file into the production tenant.
   During the upload process, the system runs compatibility checks on the solution to ensure the following:
   - The tenant where you develop your solution and the tenant where you upload your solution must be on the same release.
   - SAP does not import a patch to its cloud solution at the same time.

   During the upload process, the status of the solution is *In Deployment*. If no errors or warnings are found during upload, you can activate the solution.
4. In the *Implementation Manager*, click the *Activate* button.
   The status of the solution changes to *Deployed*.

Re-trigger Automated Distribution

Once a solution has been successful deployed and exists in a lower version in other tenants in the same system, then you can use the automated distribution function to restart the distribution process in those tenants. The system triggers the upload and activation of the solution in those tenants through the **SAP_SYSTEM** user, and not with the service agent user.

You can restart the process by navigating to the *SAP Cloud Applications Studio* > *Implementation Manager* > *Tenant Status* > *Automated Distribution*. You can also check the *Information Area* for more details on the status of the distribution of the tenants.

The distribution of solutions can be unsuccessful due to two reasons as shown below:

1. **Both the upload and activation of the solution had failed**: In this case, you must trigger a restart of the upload of the solution by clicking on *Automated Distribution*. Both the upload and activation of the solution is automated.
2. **Only the activation of the solution has failed**: In this case, you must login to the relevant tenant and manually activate the solution.

For both the cases, you must check the version history logs for the respective tenants.

Create a Patch for a Solution

If you need to make changes or corrections to a customer-specific solution after you have assembled it, you can create a patch. Depending on the tenant on which you are working, the process of creating a patch is different:

- Create a Patch on a Customer's Tenant [page 152]
- Create a Patch on Your Development Tenant [page 157]

Enable and Disable Your Solution in the SAP Cloud Solution

You use this function, for example, when you have created a patch solution on the customer's test tenant and want to test the patch solution in the SAP cloud solution.

For more information, see Enable and Disable Your Solution in the SAP Cloud Solution [page 182].

Create a Copy of a Solution

You can use a copy of your solution, for example, to continue developing in order to try out different scenarios without disrupting the original. For doing this, you can download a copy of an existing solution and then upload this copy to the same tenant or a different tenant.

> **i Note**
>
> You cannot merge the content of the copy back to the original solution. After having achieved a satisfactory development status of the copy, you need to repeat all changes in the original solution that you made in the copy.

1. In the studio, log on to the tenant where you created the solution and open the solution.
2. In the *Solution Explorer*, right-click the solution and select *Activate*.
3. In the *Solution Explorer*, right-click the project and select *Check In All Files*.
4. In the *Implementation Manager*, click the *Assemble and Download* button and then *Download a Copy* .
5. Enter a description for the copy and save the `.zip` file locally.
   The status of the original solution remains *In Development*.
6. Optional: If you want to upload the copy to a different tenant, log on to this tenant.
7. In the *Implementation Manager*, click the *Upload* button and select the `.zip` file of the copy.
8. Agree to the terms of the upload disclaimer and click *Continue*.
   The copy is uploaded and opened. The status of the solution is *In Deployment*.
9. In the *Implementation Manager*, click the *Activate* button.
   When the system has activated the solution, you can create a patch and continue developing. For information about patches, see here [page 178].

Import Items of a Solution Template

After you have downloaded a solution template, you can import this template to add its items to any of your solutions.

For more information, see Solution Templates Quick Guide [page 171].

View Where-used List of Extension Fields

You can get a list of content where an extension field has been used. The list includes the following content types:

- Business roles
- Extension scenarios
- Code list restrictions
- Mashups
- Page layouts and adaptation
- Workflows
- BRF rules

You can also download the extracted content into an excel file.

> **i** Note
>
> Analytics content such as data source, reports, and so on are not available in the where-used list of extension fields.

Troubleshooting

If you were not able to activate the solution or patch, try the following:

1. In the *Implementation Manager*, click *Show Log File* to display updated log information on the *Version History* tab.
   If your solution or patch did not contain any errors, click the *Activate* button to activate the solution or patch again.
2. If this does not work, click the *Reactivate* button in the *Implementation Manager* to generate all the objects in the solution or patch again.
3. If reactivation fails, create an incident. For more information, see Report an Incident [page 628].

Further Tasks

Users with the **Studio Administrator** role can use the *Administration* toolbar to access a number of administrative tasks, such as creating and deleting customer-specific solutions or solution templates and managing user sessions.

For more information, see Administration Quick Guide [page 159].

6.6.3 Migration Quick Guide

If you open a solution or solution template that has been created in a studio version older than 1302, the *Migrate Solution* button is displayed in the *Solution Explorer*.

Business background

The reason for this is that a new scripting language compiler has been introduced. It offers the following advantages:

- Improves performance, for example, reduces activation time
- Use functions developed for new compiler
- Enables use of SAP HANA

During migration, the following changes occur:

- Multiplicity syntax. For more information, see Node (Business Object) [page 211].
- Target annotation in associations. For more information, see Association (Business Object) [page 196].
- Generated runtime objects

Important tasks

Do I need to migrate my solution?

No, you do not need to migrate your solution, and you can still use the solutions and solution templates you created with the old compiler.

However, you will not benefit from the advantages of the new backend compiler. In addition, since studio version 1302, new functions have been developed for the new compiler that you cannot use with your old solutions and solution templates. Please also note that you must migrate old solution templates, if you want to use them in a new solution.

Solution or solution template created in this studio version	What you need to do to benefit from the new compiler
SAP Business ByDesign Studio 4.0	Migrate your solution or solution template
SDK for SAP's On-Demand Solutions 1.0	Migrate your solution or solution template
SDK for SAP's On-Demand Solutions 1.5 (1211)	Migration not necessary

Solution or solution template created in this studio version	What you need to do to benefit from the new compiler
SAP Solutions OnDemand Studio 1302, 1305	Migration not necessary
SAP Cloud Applications Studio August 2013 (1308)	Migration not necessary

> → Tip
>
> To check the compiler version of your solution, open the *Properties* window. The *Compiler Version* field typically contains two numbers separated by a slash, for example, 1302 / 1211. The first number indicates the compiler version of the `.absl` files. The number after the slash indicates the compiler version of the `.bo` and `.xbo` files.

How do I migrate my solution to the new scripting language compiler?

1. Open your solution or solution template in the new studio version.
   In the *Solution Explorer*, the *Migrate Solution* button appears.
2. Click the *Migrate Solution* button to migrate your solution or solution template.

   > i Note
   >
   > The migration cannot be reverted and may take some time.

   > → Tip
   >
   > The *Migrate Solution* option is also available in the *Administration* menu. It is grayed if your solution already uses the current scripting language compiler.
   >
   > If the solution has errors, these are displayed in the *Error List*. Resolve the errors and click the *Migration Solution* button again.

   This is what happens during the migration:
   - All files of your solution or solution template are checked out so that the solution cannot be changed during the migration process.
   - The entire solution is activated.
   - The business object files are modified.
   - The *Output* window shows you what has been changed.
   - The compiler version is changed and the whole solution is activated automatically.
3. Check in all files, test your solution, and click *Assemble and Download Customer-Specific Solution*.
   In the *Solution Explorer*, the *Migrate Solution* button disappears and, in the Administration menu, the *Migrate Solution* option is grayed out.

   > i Note
   >
   > - If the migration was not successful, you cannot start the migration process again. Please create an incident instead and ask SAP Support for help.

- Once the solution has been migrated, the migration cannot be reverted.

4. Deploy your migrated solution to the target tenant.
   The runtime objects in the target tenant will be migrated when you activate your solution there. If you migrate data, this may take some time.

Which limitations apply?

- You can only migrate your solution if it is `In Development` or if it is a patch solution.
- If your old solution contains alternative key elements that use data types with unrestricted length, for example, Identifier or LANGUAGEINDEPENDENT_Text, you cannot migrate your solution. However, you can replace these data types with compatible data types. This is also possible in patch solutions.
  This is an overview of the data types that you need to replace and of the data types that you can use instead:

  > **i Note**
  >
  > You only need to replace these data types, if you have used them in *AlternativeKey* elements.

Data types that you need to replace	Data types that you can use instead
BASIS.Global:Identifier	AP.PDI.bo:EXTENDED_ID
AP.Common.GDT:Identifier	AP.PDI.bo:EXTENDED_ID
BASIS.Global:Name	AP.Common.GDT:EXTENDED_Name
AP.Common.GDT:Name	AP.Common.GDT:EXTENDED_Name
AP.Common.GDT:LANGUAGEINDEPENDENT_Name	AP.Common.GDT:LANGUAGEINDEPENDENT_EXTENDED_Name
AP.Common.GDT:Description	AP.Common.GDT:LONG_Description
AP.Common.GDT:Note	AP.Common.GDT:LONG_Description
AP.Common.GDT:Text	AP.Common.GDT:LONG_Description
AP.Common.GDT:LANGUAGEINDEPENDENT_Text	AP.Common.GDT:LANGUAGEINDEPENDENT_EXTENDED_Text

- You cannot migrate your solution in the following case:
  1. if your old solution contains a business object node that has the same name as another business object in the solution **and**
  2. if your solution contains associations to the business object or the business object node.

However, you can resolve this ambiguity by adding the name of the `Root` node or the name of the business object to the association target. Here is an example:

Example

This is the target business object...	Do not use this source business object... (Wrong)	but this source business object instead... (Correct)

`businessobject Target`
`{`
`element ID:ID;`
`}`
	`businessobject Source`
	`{`
	`element ID:ID;`
	`node Target[0..n] {`
	`element TargetID:ID;`
	`}`
	`association`
	`FirstTarget to Target;`
	`}`
		`businessobject Source`
		`{`
		`element ID:ID;`
		`node Target[0..n] {`
		`element TargetID:ID;`
		`}`
		`association`
		`FirstTarget to`
		`Target.Root;`
		`OR`
		`association`
		`FirstTarget to`
		`Source.Target;`
		`}`

Wrong: The `Target` identifier following the `to` keyword of the `FirstTarget` association is not unique.

Correct: Specify whether the association points to the

- root node of the other `Target` business object (association `FirstTarget to Target.Root`) or to the
- `Target` subnode of the same business object (association `FirstTarget` to `Source.Target`)

What happens if I choose not to migrate my solution?

If you do not migrate your solution, you can continue to use it at it is. However, you will not benefit from the advantages of the new backend compiler. Please note that you must migrate old solution templates, if you want to use them in new solutions.

6.7 Solutions, Solution Templates, and Patches

6.7.1 Solution Templates Quick Guide

You can create templates containing reusable items and upload these items to customer-specific solutions.

For more information, see Solution Templates [page 174].

Tasks

Create a Solution Template

You can create templates containing reusable items and upload these items to customer-specific solutions.

For more information, see here [page 175].

Create a Copy of a Solution Template

You can use a copy of your template, for example, to continue developing in order to try out different scenarios without disrupting the original. For this purpose, you download a copy of an existing template and then upload this copy to the same tenant or a different tenant.

> **i** Note
>
> You cannot merge the content of the copy back to the source template. After having achieved a satisfactory development status of the copy, you need to repeat all changes in the original that you made in the copy.

1. In the studio, log on to the tenant where you created the template and open it.
2. In the *Implementation Manager*, click the *Assemble and Download* button and then *Download a Copy* .
3. Enter a description for the copy and save the `.zip` file locally.
   The status of the original remains *In Development*.
4. Optional: If you want to upload the copy to a different tenant, log on to this tenant.
5. In the *Implementation Manager*, click the *Upload* button and select the `.zip` file of the copy.
6. Agree to the terms of the upload disclaimer and click *Continue*.
   The copy is uploaded and opened. The status of the template is *In Development* and you can continue developing.

Download a Solution Template

When you have completed your template, you need to download the template before you can import its items to a solution.

1. In the studio, log on to the tenant where you created the template, and open the template.
2. In the *Solution Explorer*, right-click the template and select *Check In All Files*.
3. On the *Implementation Manager* toolbar, click the *Assemble and Download* button.
4. Enter a file name to save the template as a `.zip` file locally.
   The system checks the template for consistency and then downloads all the items in the template to the `.zip` file.

Import Items of a Solution Template

After you have downloaded a solution template, you can import this template to add its items to any of your solutions.

1. Log on to the tenant where you want to import the items of your template to a specific solution.
2. Create a new customer-specific solution or open an existing solution. Please note that the solution must have the status *In Development*.

   > **i** Note
   >
   > Please note that the solution template and the solution, into which you import the items of the solution template, must have been created for the **same** deployment unit. However, if you have created your solution template in the Foundation deployment unit, the solution can have any deployment unit.

3. In the *Implementation Manager*, click the *Import Solution Template* button and select the `.zip` file of the template that you have downloaded before.
   The items of the template are added to the solution.

   > **i** Note
   >
   > If the solution and the template contain items that have the same name, the system replaces the items in the solution by that of the template.
   >
   > Please note the following exceptions:
   >
   > - If an item with the same name exists in another folder of the solution, the system doesn't import the item
   > - If the template contains a BAC element and the solution already has a BAC element, the system doesn't import the BAC element of the template.

Make sure that your solution works well with the replaced or added items, for example, check if the associations are still valid. When you have finalized your solution, activate it.

Upload a Solution Template to Another Studio Version

If you want to continue developing your solution template in another studio version, for example, to make use of new functionality, you need to upload the solution template to that studio version. As a prerequisite for this, you have downloaded your solution template from an older studio version that is compatible with the new studio version. For information about the compatible versions, see Upgrade Information [page 17].

> **i** Note
>
> Please make sure that the SAP cloud solution from which you have downloaded the solution template matches the SAP cloud solution to which you want to upload the solution template.

To upload your solution template, proceed as follows:

1. Open the studio version to which you want to upload the solution template and log on to the tenant.
2. On the *Implementation Manager* toolbar, click the *Upload* button.

3. Select the `.zip` file of the downloaded solution template.
   An upload disclaimer appears.
4. Agree to the terms of the upload disclaimer and click *Continue*.
   The system unpacks the `.zip` file into the tenant and opens the solution template in the *Solution Explorer*.
   You can now continue working on your solution template.

6.7.2 Solution Templates

With a solution template, you can organize development content that you want to reuse for customer-specific solutions. Solution templates enable you to easily start the development of customer-specific solutions, for example, for a specific industry.

What you can do with a solution template

In a solution template, you can create all items that you want to reuse in a solution, for example, business objects, actions, and events. When you have completed your template, you must download it to be able to import it into a solution. A template always has the status *In Development* and therefore you don't need to create a patch of a template.

When you create a new solution or open an existing solution or a patch, you can import the items of your template into this solution or patch. For more information, see Solution Templates Quick Guide [page 171].

> **i** Note
>
> To create a solution template, you must log on with a user that has the studio administrator role.

What you can't do with a solution template

- You can't switch on the admin mode for a solution template. Therefore you can't, for example, define analytics content or create a mashup in a solution template.
  For information about the admin mode, see Working in the Integrated Development Environment [page 42] under **Administration Menu**.
- You can't create BC sets using SAP BCOs in a solution template.
- If you create a business object extension in a solution template, you can't use the following functions:
  - Enhance Enterprise Search
  - Enhance Forms
  - Enhance Report
  - Enhance Screen

What you need to know when you import a solution template

Please note the following:

- The solution template and the solution, into which you import the items of the template, must be based on the same cloud solution from SAP. For example, you can't import items from a template created for SAP Business ByDesign into a solution for SAP Cloud for Customer.
- The solution template and the solution, into which you import the items of the solution template, must have been created for the **same** deployment unit. However, if you have created your solution template in the Foundation deployment unit, the solution can have any deployment unit.

You can import a solution template into the studio version in which it has been created and into the two subsequent versions, for example:

If you have created your solution template in this version ...	... you can upload it to this version	... and to this version
SAP Solutions OnDemand Studio 1302 (February 2013)	SAP Solutions OnDemand Studio 1305 (May 2013)	SAP Cloud Applications Studio 1308 (August 2013)

When you import a template into a solution, all items of the template are added to the solution. If an item in the solution has the same name as in the template, the system replaces the item in the solution with that of the template.

Please note the following exceptions:

- If an item with the same name exists in another folder of the solution, the system doesn't import the item.
- If the template contains a BAC element and the solution already has a BAC element, the system doesn't import the BAC element of the template.

6.7.3 Create a Solution Template

Overview

You can create templates containing reusable items and import these items into customer-specific solutions. This allows you to easily start the development of customer-specific solutions by reusing common development content, for example, for a specific industry.

For more information, see Solution Template [page 174].

Prerequisites

You are logged on to the repository of your development tenant or to the repository of your customer's test tenant with a user that has the studio administrator role.

Procedure

1. On the *Administration* toolbar, click *Create Solution* ( ).

   The *Create Solution* dialog box opens.
2. Select the *Solution Template* type.
3. Enter the customer and a description for the solution template.
4. Select the deployment unit in which you want to create your template and click *OK*.
   For more information about deployment units, see Deployment Unit (Business Object) [page 205].
5. Enter your contact information and click *OK*.
   Your solution template is created.

> **i Note**
>
> If you want to change the partner contact information, please do the following: In the *My Solutions* window, right-click the solution and select *Solution Properties*. The *Solution Properties* dialog opens, where you can change the contact information. Click *OK* to save your changes.

Result

You have created a solution template.

How your solution template is displayed

- In the *My Solutions* window, the new solution template is displayed as follows: [solution template description] [solution template name].
- In the *Solution Explorer*, the solution template is opened and displayed as follows: [solution template description] (In Development).

What you can do with your solution template

You can create all items that you want to reuse in a solution, for example, business objects, actions, and events. When you have finalized your template, you can download it. You can then create a new solution or you can open an existing solution or patch and import the items of your template to this solution or patch. For more information, see Solution Templates Quick Guide [page 171].

> **i Note**
>
> When you import a template into a solution, all items of the template are added to the solution. If an item in the solution has the same name as in the template, the system replaces the item in the solution with that of the template.

6.7.4 Create a Solution

You can design and develop customer-specific solutions to enhance SAP's standard cloud solution for a specific customer. In the studio, you can create a solution in which all items that you develop or changes that you make to existing entities are stored.

Depending on the tenant that you are working on, the following applies:

- **Customer's Tenant**
  You create your solution for this specific customer on the customer's test tenant.
- **Development Tenant**
  On your development tenant you can develop solutions for different customers. Each time you create a solution, ensure that this solution is assigned to the correct customer ID. For more information, see Switch a Customer Assignment [page 158].

> **i Note**
>
> To create customer-specific solutions, you must be logged on to the studio with a user that has the **Studio Administrator** role.

Procedure

1. Optional: If you are developing on your development tenant, in the *Administration* menu, choose *Switch Customer*.
   Check that the ID of the customer for which you want to develop the solution is displayed. If not, select a different customer or create a new customer assignment.

   > **i Note**
   >
   > If you are logged on to your development tenant and you are creating a new solution, the system automatically creates the solution for the customer that is currently displayed in the *Switch Customer* dialog box. Be aware, that only a solution with the correct customer ID can be uploaded to the corresponding tenant of your customer.

2. On the *Administration* toolbar, click *Create Solution* ( ).

   The *Create Solution* dialog box opens.

   > **→ Tip**
   >
   > You can also use this function to create a solution template. For more information, see Create a Solution Template [page 175].

3. Select the *Customer-Specific Solution* type.
4. Enter the customer and a description for the solution.
5. Select the deployment unit in which you want to create your solution and click *OK*.
   For more information about deployment units, see Deployment Unit (Business Object) [page 205].
6. Enter your contact information and click *OK*.
   Your solution is created.

> **Note**
>
> If you want to change the partner contact information, please do the following: In the *My Solutions* window, right-click the solution and select *Solution Properties*. The *Solution Properties* dialog opens, where you can change the contact information. Click *OK* to save your changes.

Result

You have created a solution.

How your solution is displayed

- In the *My Solutions* window, the new solution is displayed as follows: [solution description] [solution name].
- In the *Solution Explorer*, the solution is opened and displayed as follows: [solution description] (In Development).

What you can do now

You can now create a business object or use the other functions of the developer desktop. For more information, see Business Objects Quick Guide [page 398] and Overview of the Developer Desktop [page 360].

Related Information

Mobile Solutions [page 774]

6.7.5 Patches

Overview

A patch is needed if you want to make changes or corrections to a customer-specific solution after it has been assembled. The customer-specific solution is then in maintenance mode and you can only make restricted changes for certain content types. For more information, see Maintenance Mode [page 185].

> **Note**
>
> Only users with the **Studio Administrator** role can create patches.

Patch Types

Depending on the type of customer's tenant where you are developing the creation and handling of a patch works differently:

- **Creating a Patch on a Customer's Test Tenant**
  When you create a patch of a solution on a customer's test tenant, the system creates a copy of your solution in a new namespace allowing you to make and test your changes without disrupting the original solution. This patch type is called a patch solution. When you want to test your patch solution in the test tenant, you first need to use the *Implementation Manager* in the studio to enable the patch solution instead of the original solution in the test tenant.
  When you assemble the patch solution, the patch is copied back to the namespace of the original solution. The patch solution is used for all future updates to the original solution. Any changes you need to make in the future can be made by creating a new patch again. To do this, open the existing patch solution in the *Implementation Manager* toolbar and click *Create Patch*. All future patches are created in the same patch solution but each patch has a different version number.
  For more information about which steps to perform to create a patch solution, see Create a Patch on a Customer's Test Tenant [page 152].
- **Creating a Patch on Your Development Tenant**
  When you create a patch on your development tenant, the system creates this patch in the same solution and namespace but with a different version number. For more information about which steps to perform, see Create a Patch on Your Development Tenant [page 157] .

Deletion Patch

A deletion patch helps you to delete elements that are not in use, resulting in better performance and removal of obsolete data. It allows deletion of delivered content even in maintenance mode. This patch follows a similar lifecycle of a regular patch, that is, it would have the same patch namespace, the same versioning, and would follow a similar assembly and deployment lifecycle management process. For more information, see Deletion Patch [page 180]

Distribution of Patches in the Production System

After you have activated a patch, the solution including the patch is available for productive use by the customer.

If the solution is already running on other tenants in the same system as the production tenant, the patch is uploaded and activated on these tenants automatically. This happens in the background after the patch is successfully activated in the production tenant and may take some time. If the solution is running on other tenants in different systems, you need to upload the patch to these tenants manually.

To find out for which tenants the patch is intended, in the *Implementation Manager*, select the *Tenant Status* tab.

Assembly of Patch Solutions with New and Changed Items

You can assemble files that are new or changed in a patch.

The items are part of a zip file. and the technical consistency of the zip file is ensured by running the assembly process for the dependent files.

> ℹ Note
>
> - Partial assembly is available only for patch solutions.
> - It's mandatory to deploy each partially assembled patch solution – you cannot skip the deployment of a patch version.

Handling Errors in Lifecycle Management

Exceeding Extension Field Limit [page 181]

6.7.5.1 Deletion Patch

Remove extension fields in an extension object (XBO) from a deployed solution.

To remove the fields, you must create a deletion patch. This helps you to delete elements that are not in use, resulting in better performance and removal of obsolete data. A deletion patch allows deletion of delivered content even in maintenance mode. This patch follows a similar lifecycle of a regular patch, that is, it would have the same patch namespace, the same versioning, and would follow a similar assembly and deployment lifecycle management process.

Prerequisites

- You have authorization to the *Deletion Patch* work center view.
- The solution is of the customer-specific solution type.
- The solution is in *Assembled* status as of the February 2019 release, and one regular patch should have already been created. This means that the deletion patch cannot be the first patch.

Procedure

1. On the *Implementation Manager* toolbar, click the *Create Patch* button. The system displays a disclaimer pop-up in a dialog box stating that the deletion of objects in this patch would also delete the data corresponding to these objects in the development tenant as well as on the tenants where this patch is deployed.

2. Click *OK* in the message dialog box. The solution status changes to *In Development – Deletion Patch*.
3. Delete the extension fields in an XBO.
4. In the *Solution Explorer*, right-click your solution and choose *Activate*.
   Once the XBO is activated, adjust these extension fields in other content types to make the solution consistent. In order to decide which content types need to be adjusted, run a solution level consistency check and make the adjustments accordingly.
5. Assemble and download the deletion patch and then deploy it to the target tenants.

> ⚠ Caution
>
> If you remove extension fields, the data for the extension fields will be lost permanently.

> i Note
>
> - You cannot create a deletion patch in an original solution - it can only be created on a standard patch. This patch is valid from version 4 and maximum three consecutive deletion patches are allowed.
> - You cannot skip the deployment of a deletion patch. For example, if you have a deletion patch (version 5) and a normal patch (version 6), you cannot deploy version 6 patch without deploying the deletion patch.
> - You cannot create a deletion patch while another patch is in the in-development status
> - You cannot make any other changes in the BO definition, such as, adding a new field, changing field labels, adding annotations. We recommend that you do not remove a field and activate the deletion patch if you are planning to revert the change.
> - If the extension fields are used in ABSL scripts, reports, form templates, web service and so on, then you will get a consistency check error. You must remove the fields from the ABSL scripts, or reports, or forms before removing it from the extension object.
> - Import of solution template into a deletion patch is restricted.
> - Download a copy as new solution in a deletion patch is restricted.
> - You cannot delete personal data and sensitive personal data in a deletion patch.
> - If an extension field is used to extend a standard entity (such as screens, web services, data sources, reports), deletion of the extension field removes the reference from all these entities automatically. Only in the case of standard forms, if you have manually added the extension field to a form template from the UI, then you must remove this reference manually from *Administrator* > *Business Flexibility* >> *Form Template Maintenance*.

6.7.5.2 Exceeding Extension Field Limit

Message	Number of elements in the &1 node has reached the maximum limit.

Description	Using PDI or KUT in any client, if you create more than or equal to 744 extension fields in a standard BO node, the system displays an error message. For example, in client ABC, in the ROOT node of SERVICE_REQUEST BO, 200 KUT fields are created. In the same BO node, if there are already three PDI solutions with 200 extension fields, the system adds all the fields for the BO and displays an error message. In the **original solutions** that do not have a patch in the system, this works differently as it keeps a buffer of the fields that would be created in patch. So, if the extension fields limit in the original solution exceeds half of the maximum limit (744/2 = 372), then the system throws an error during activation. The error message is displayed because the extension fields get doubled when the patch is created.
Solution	• Delete the extension fields created for the same BO-NODE combination in the current solution that displays the error. • Delete the KUT fields or extension fields from other PDI solutions created for the same BO-NODE combination. • Delete the unused PDI solution created for the same BO-NODE combination.

6.7.6 Enable or Disable Your Solution in the SAP Cloud Solution

When you create a new solution on your tenant, this solution is automatically enabled for your business user in the SAP cloud solution. This means that your solution is visible in the SAP cloud solution when you log on with your business user on this tenant.

Enable or Disable Your Solution on Your Development Tenant

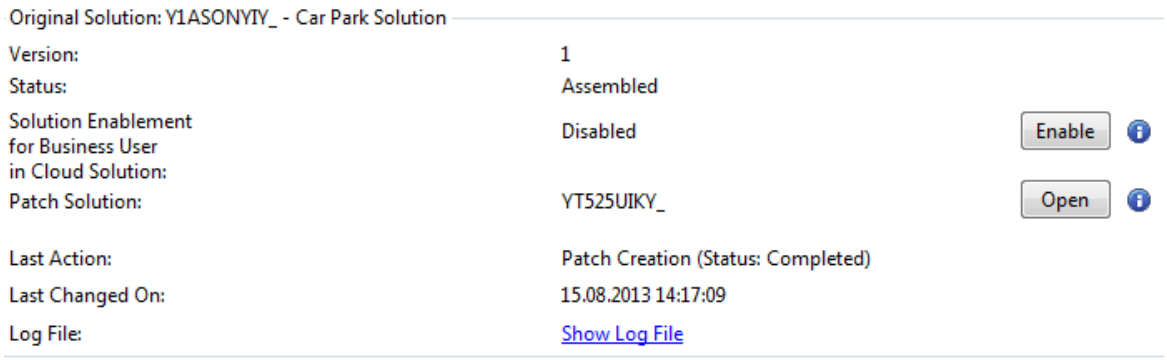
When you are developing several solutions in parallel on your development tenant, you may want to disable a solution for your business user in the SAP cloud solution, for example, if this solution affects the solution that you want to test. To do this, open the solution you want to disable and in the *Implementation Manager*, click *Disable*.

Enable or Disable Your Solution on the Customer's Test Tenant

When you create a patch for your solution, the original solution is still enabled for your business user. To test your patch solution, you need to enable it instead of the original solution. This means that the system automatically removes the scoping of the original solution and disables the original solution. After logging on to the SAP cloud solution, you will have to scope your patch solution before you can test it.

Procedure for Solutions on the Customer's Tenant

1. In the *Implementation Manager*, on the *Current Solution* tab, check whether the original solution or the patch solution is open:



Solution details on the Current Solution tab of the Implementation Manager

   The title of this area of the screen shows which solution is currently open, the original solution or the patch solution (if a patch solution exists). In this example, the original solution is open.
2. In the *Solution Enablement for Business Users in Cloud Solution* field, click *Enable* or *Disable*.
   You use the *Enable* function to make either the original solution or the patch solution visible in the SAP cloud solution. In the example, if you open the patch solution first and then use the *Enable* function, you make the patch solution visible and the original solution is automatically removed and disabled.
   You use the *Disable* function for special scenarios, for example, if your customer has opened a change project on the test tenant before you created the patch solution. If you want to know more, see What You Need to Consider When a Change Project Is on the Same Test Tenant [page 154].

> **i Note**
>
> If a solution template is being developed on the customer's test tenant or has been uploaded to this tenant, it is also automatically enabled. If there is a change project on the same tenant, you to need to disable the solution template before the change project can be merged to the production tenant.

   Check this table to see what happens when you enable or disable a solution:

What do you want to do?	What happens in the SAP cloud solution?	What else do you need to know?
Enable the patch solution in the SAP cloud solution on the test tenant.	The system removes the scoping of the original solution in the SAP cloud solution and disables the original solution automatically.	After logging on to the SAP cloud solution with your business user, you need to scope your patch solution manually.
Enable the original solution in the SAP cloud solution on the test tenant.	The system removes the scoping of the patch solution in the SAP cloud solution and disables the patch solution automatically.	After logging on to the SAP cloud solution with your business user, you need to scope your original solution manually.

What do you want to do?	What happens in the SAP cloud solution?	What else do you need to know?
Disable the patch solution in the SAP cloud solution on the test tenant.	The system removes the scoping of the patch solution in the SAP cloud solution.	
Disable the original solution in the SAP cloud solution on the test tenant.	The system removes the scoping of the original solution in the SAP cloud solution.	

→ Tip

If you need to disable both the original solution and the patch solution, first disable the solution that is open. Then open the other solution and disable it. It doesn't matter which solution is open first.

6.8 Bug Fix in Production and Pre-production Tenants

Overview

Test tenants are upgraded two weeks before the production tenants are upgraded, and test tenants are on a higher release for two weeks. During this period, if a bug is detected in the production tenant from a solution, there are no mechanisms to create and deploy a patch from the test to the production tenant, as it is not allowed to deploy a patch from a higher to a lower release. With this feature you can change the .absl files directly in the production or pre-production tenants to overcome the reported issues.

Prerequisites

- The work center view *Production Fix Authorization* is already assigned to the Studio developer.
- The solution is scoped with the **Deployed** status.

ℹ Note

Please take into account the following points before you make any changes:

- Changes are allowed only to .absl files. No other content is supported.
- You cannot create or delete already created .absl files.
- You cannot change .absl files belonging to data disclosure items.
- This is supported only for customer-specific solutions. Multi-customer solutions and key-user solutions are not supported.
- During the duration when you are making the bug fix (Status of the solution is Deployed — In Correction), the solution is locked. Hence you cannot upload or deploy a patch solution. You also cannot perform any tenant operation, such as tenant move or copy.

- There is no automatic double maintenance of the code changes to the upgraded test tenant. You must manually make the corrections in the test tenant which has been upgraded to the higher release.
- The following are additional checks applicable for Pre-production tenants:
  - If the production tenant with this solution is available on the same system as the pre-production tenant, then bug fixing is not allowed
  - If the test tenant with this solution is available on the same system as the pre-production tenant, then it's allowed with a warning – the code change has to be manually maintained in the other test tenant

Process Flow

1. In the *Solution Explorer*, click to open a correction patch. The solution is available for editing and the status of the solution is changed to **Deployed - In Correction**. Read the Disclaimer pop-up and agree to proceed with the code changes.
2. Implement the required .absl changes, by using *Check Out*, *Check In*, *Activate* and *Revert* actions in the .absl file:
   1. Check out the specific .absl file for which the code change needs to be carried out.
   2. Perform the .absl code changes as required using the Studio absl editor.
   3. Save and activate the modified .absl file. On successful activation, the code changes are active for the Business users.
   4. Check in the activated .absl file.
3. Once you finish making the changes, click to close the correction patch. The status of the solution is changed to **Deployed**.
4. Test your solution. Right-click on the .absl file and click on *Revert*. This .absl file goes back to the previous checked in and activated version.
5. Manually make the same corrections to the solution in the development tenant.

6.9 Maintenance Mode

Overview

A customer-specific solution is in maintenance mode, if it has been assembled and downloaded; the solution status then is *Assembled*. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.

Deletion

When your solution is in maintenance mode, you can delete the following items:

Item	Where you can find further information
Business Configuration Object (BCO)	Business Configuration Quick Guide [page 362]
Business Configuration Set (BC Set)	
Business Configuration View (BC View)	
OData Service Extension	Create an ODATA Service Extension [page 463]
Code List Data Type	Create a Code List Data Type [page 472]
Embedded Component	Model an Embedded Component [page 716]
Screens	Generate Screens for a Business Object [page 429]
Report	Create a Report [page 534]
Script File (ABSL)	Actions, Events, and Validations [page 403]
Web Service	Web Services Quick Guide [page 586]
Mass Data Run Object	Mass Data Run Quick Guide [page 537]
UI Switch	Flexible UI [page 760]
Tax Decision Tree	Create a Tax Decision Tree [page 394]
References to customer-specific fields	Create References to Customer-Specific Fields [page 453]
Reference to a joined data source created by an administrator in the cloud solution	Create a Joined Data Source [page 532]
Reference to a combined data source created by an administrator in the cloud solution	Create a Combined Data Source [page 530]
Reference to a key figure created by an administrator in the cloud solution	Create a Key Figure [page 535]

Related Information

Maintenance of Business Configuration Content [page 372]
Maintenance of Analytics Content [page 526]
Maintenance of Business Objects [page 414]
Maintenance of Business Object Extensions [page 441]

Patches [page 178]

7 Scripting Languages

7.1 Overview of Scripting Languages

Overview

Scripting Languages encompasses the following reference material:

Get Informed About Scripting Languages for the Studio

- Scripting Language Reference [page 189]
  The scripting language is used to define business objects and business object extensions as well as to implement the business logic for cloud partner solutions.
  This lightweight language is easy to learn and to use with its focused set of features.
  The keywords, methods and functions of the scripting language are described in separate sections according to the task you want to perform:
  - Syntax for Business Object Definitions [page 191]
    You use these keywords and functions to define the structure and interface of business objects.
  - Syntax for Business Object Extension Definitions [page 218]
    You use these keywords and functions to define the structure and interface of business object extensions.
- Syntax for the Implementation of Business Logic [page 235]
  You use these methods and keywords to implement the business logic of your solution. For this task, you can also use predefined built-in functions and reuse services:
  - Built-In Functions [page 320]
  - Basic Reuse Libraries [page 274]
  - Reuse Libraries for Business Areas [page 308]
- Basic Data Types [page 329]
  The scripting language supports basic data types such as string, numeric and Boolean values. In addition, the existing metadata of the SAP underlying cloud platform for core data types (CDTs) and global data types (GDTs) is made available.

Get Informed About Scripting Languages for the UI Designer

- Front-End Script for the User Interface Designer [page 335]
  The front-end script is the programming language used in the UI designer to define a set of rules and the logic of the behavior of UI controls. For example, you can write a front-end script to enable an edit button only when a field is selected on the user interface.

> ⚠ Caution
>
> Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

> **i Note**
>
> You can also refer to SAP Cloud Application Studio Performance Best Practices for more information on some best practices to keep in mind while creating ABSL scripts.

Availability

> **⚠ Caution**
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

7.2 Scripting Language for the Studio

7.2.1 Scripting Language Reference

Overview

The scripting language is used to define business objects and business object extensions as well as to implement the business logic for cloud partner solutions. This lightweight language is easy to learn and to use with its focused set of features.

The sandbox environment for the scripting prevents a partner who has created erroneous implementations from damaging data of other users by accident. This kind of sandbox environment is ensured by the script execution runtime. Security-relevant features, such as direct access to database tables, are strictly controlled or not made available in the scripting language. Instead, all data access is made available through the business object infrastructure.

The keywords, methods and functions of the scripting language are described in separate sections according to the task you want to perform:

- Syntax for Business Object Definitions [page 191]
  You use these keywords and functions to define the structure and interface of business objects.
- Syntax for Business Object Extension Definitions [page 218]
  You use these keywords and functions to define the structure and interface of business object extensions.
- Syntax for Implementation of Business Logic [page 235]
  You use these methods and keywords to implement the business logic of your solution.

For this task, you can also use predefined built-in functions and reuse services. For more information, see the following documentation:

- Built-In Functions [page 320]
- Basic Reuse Libraries [page 274]
- Reuse Libraries for Business Areas [page 308]

To fully understand the keyword descriptions, it is essential to have a basic understanding of the business object model and its features, such as associations, actions, and queries.

> **i** Note
>
> Model names of SAP entities are in British English according to ISO 11179.

Code Completion

Based on the code you type, the code completion feature presents you with a list of options. For example, if you type a period (.), which separates the members in a path expression, the code completion is automatically activated and a drop-down list with the potential options is displayed. You can also use key combinations to display the valid code completion options, for example, `CTRL` + `SPACEBAR`. For more information, see the relevant syntax documentation under **Code Completion**.

Syntax

The syntax conventions for scripting language are as follows:

Character	Description
[ ]	Indicates that you can use none, one, or more of the enclosed options. Do not include the brackets in your option.
|	A vertical line between two parts of a statement indicates that only one of these two parts can be used within the statement.
?	Indicates that the symbol (or the group of symbols in parenthesis) to the left of the operator is optional (it can appear just once or not at all). You can use it as a postfix to "[ ]".
+	Indicates that something can be repeated at least once or any number of times. You can use it as a postfix to "[ ]".
*	Indicates that something can appear any number of times (and possibly be skipped altogether). You can use it as a postfix to "[ ]".

Character	Description
XXX	Indicates a placeholder. Replace placeholders with actual values in your code snippet.

7.2.2 Business Object Definitions

7.2.2.1 Syntax for Business Object Definitions

Use the keywords and functions listed below to describe the structure and interface of a business object in the studio.

For more information about the general syntax conventions and rules, see Scripting Language [page 189].

Code Completion

You can use **CTRL** + **SPACEBAR** at any cursor location to display the valid code completion options for the current context . If there is only one option, the code is completed at once. The code completion is also activated, if you enter a period (.) in a path expression or a colon (:), for example, in an *element* statement.

Naming Rules in Business Object Definitions

Comply with the following naming rules:

- The name of an entity must start with a letter and can contain letters, digits, and underscore characters. Do not start a name with “SAP_”.
- Related entities within a business object, for example, elements, associations, and actions, cannot have the same name as the business object. This does not apply for messages.
- Do not use the following technical terms as an entity name:
  - action
  - association
  - businessobject
  - element
  - import
  - namespace
  - node
  - query
  - to
  - type
  - using

- valuation

Comments

You can enter comments using the following syntax:

- **comment line**

  ```
  // <comment text>
  ```

  A comment line starts with two slashes (//) and spans to the end of the current line.

- **comment block**

  ```
  /* <comment text>
  <comment text>
  <comment text>
  */
  ```

  A comment block starts with a forward slash followed by an asterisk and ends with an asterisk followed by a forward slash.

List of Functions and Keywords

In the following, the functions and keywords are listed in alphabetical order:

- Action (Business Object) [page 194]
- Annotation (Business Object) [page 195]
- Association (Business Object) [page 196]
- Business Object (Business Object) [page 199]
- Default Values (Business Object) [page 202]
- Dependent Object (Business Object) [page 203]
- Deployment Unit (Business Object) [page 205]
- Element (Business Object) [page 207]
- Import (Business Object) [page 208]
- Message (Business Object) [page 210]
- Node (Business Object) [page 211]
- Raises (Business Object) [page 212]
- Valuation (Business Object) [page 215]

List of Annotations

In the following, the annotations are listed in alphabetical order:

- Access Control Context (Business Object) [page 193]
- Alternative Key (Business Object) [page 194]
- Change History (Business Object) [page 199]
- Cross Deployment Unit (Business Object) [page 201]
- Dependent Object (Business Object) [page 203]
- Deployment Unit (Business Object) [page 205]
- Extension (Business Object) [page 208]
- Language Independent (Business Object) [page 210]
- Label (Business Object) [page 209]
- Multiple Texts (Business Object) [page 211]
- Relevant For Access Control (Business Object) [page 213]
- Separately Lockable (Business Object) [page 214]
- Tooltip (Business Object) [page 214]
- Transient (Business Object) [page 214]
- With Administrative Data (Business Object) [page 217]

7.2.2.2 Access Control Context (Business Object)

Overview

Syntax	[AccessControlContext (<access context>)] `element` <element name> : <data type> ; [AccessControlContext (<access context>)] `association` <association name> `to` <target business object>;
Description	It is necessary that the access to specific instances is controlled by certain parameters. If you wish to define which elements or associations can be used to control access to certain instances, you use the `AccessControlContext` annotation. The access context can only be either **Employee** or **EmployeeOrTerritory** or **SalesOrganization**. For element, the data type must always be UUID. For association, the data type of foreign key must be UUID.
Example	[AccessControlContext (Employee)] `element` UID : UUID; [AccessControlContext (Employee)] `association` ToEmployee `to` AP.FO.BusinessPartner.Global:Employee `using` UUID ;
See Also	Define Access Control [page 424]

7.2.2.3 Action (Business Object)

Syntax	`action <name> [<Raises>]? ;`
Description	Use the *action* keyword to define an action for a node. Actions carry out the business logic and are implemented in .absl files.

Example	Definition of an action:
	`action Cancel;`
	Action definition with *raises* statement :
	`action CountHappyDays raises BonusRule.ReferenceInvalid, DateMissing;`

See also	Raises (Business Object) [page 212]

7.2.2.4 Alternative Key (Business Object)

Syntax	`[AlternativeKey] element <element name> : <data type>;`
Description	If you want to identify a node uniquely with an identifier that can be read by human beings, you use the `AlternativeKey` annotation. You can use this identifier in an association to refer to a business object node. In addition, this identifier is stored as an additional key field in the database and thus accelerates enterprise searches and queries. It is not possible to have more than one alternative key of the same type in a business object node. **i Note** Custom business objects contain an implicit alternative key of type UUID for each node, enabling the node to be used as an association target.
Example	`[AlternativeKey] element EmployeeID : ID;`

Related Information

Association (Business Object) [page 196]

7.2.2.5 Annotation (Business Object)

Syntax	`[<annotation>[(<parameter>)]?]`
Description	Annotations are used to add a category or property to an object specified by the subsequent keyword. The names of the annotations are predefined and can be displayed by the code completion. Technically, you can combine as many annotations as you want. But not all combinations are possible and not all of them make sense. For more information, see the list of possible combinations below.

Examples	`[AlternativeKey] element EmpoyeeID : ID;`
	Example with annotation and parameter
	`[DeploymentUnit(CustomerRelationshipManagement)] businessobject BonusRule { ... };`
	Example with two annotations
	`[DependentObject(TextCollection)] [MultipleTexts] node TextCollection;`

Annotation Overview

Annotation	Argument	Keyword	Can Be Combined With
AccessControlContext (Business Object) [page 193]	employee, employee or territory	element, association	—
AlternativeKey (Business Object) [page 194]	—	element	Label, Tooltip
ChangeHistory (Business Object) [page 199]	—	businessobject, node, element, association	—
CrossDeploymentUnit (Business Object) [page 201]	—	association	—
DependentObject (Business Object) [page 203]	name	node	MultipleTexts
DeploymentUnit (Business Object) [page 205]	name	businessobject	—
Description (Business Object) [page 201]	—	node	—

Label (Business Object) [page 209]	content	element	AlternativeKey, Tooltip, Transient
LanguageIndependent (Business Object) [page 210]	—	node	DependentObject(TextCollection)
MultipleTexts (Business Object) [page 211]	—	node	DependentObject(TextCollection)
RelevantForAccessControl (Business Object) [page 213]	—	association	—
SeparatelyLockable (Business Object) [page 214]	—	node	—
Tooltip (Business Object) [page 214]	content	element	AlternativeKey, Label, Transient
Transient (Business Object) [page 214]	—	element	—
WithAdministrativeData (Business Object) [page 217]	—	node, businessobject	—

7.2.2.6 Association (Business Object)

Syntax	`[ <annotation> ]? association <name> [ <multiplicity> ]? to <composition path to target node> [using <alternative key>]?;`

Description

An association is a relationship between two business object nodes. The definition of an association includes a name, the multiplicity, a target node and, optionally, a target element. The target node can be either part of the same business object or part of a different business object. You can also refer to a target node or element in a different namespace. For more information, see Import (Business Object) [page 208].

Associations are based on a foreign key relationship. The `using` clause specifies the key element of the target business object node for the relationship. As a prerequisite, the key element must be an alternative key and *foreign key usage* must be allowed for it. The `using` clause is optional. For business objects defined in the same project, the target node's hidden `SAP_UUID` element is used. For other business objects, the `UUID` of the target node is used.

The multiplicity of an association can be either [0,1] or [1,1] . If no multiplicity is specified, [1,1] is used by default. Only compositions can have a multiplicity with an upper boundary of "n". However, you can easily simulate the [1,n] multiplicity by combining a [1,n] node composition with a [1,1] intra or cross-business-object association. For more information about compositions, see Node (Business Object) [page 211].

As a rule, the target node of an association must be located in the same deployment unit as the current business object or in the Foundation deployment unit.

The path to the target node is expressed as follows: <namespace>:<business object>.<node>. The path from the root node of the business object to the target node must not be part of the expression. You can omit <namespace>, if you have imported the required namespace by using an `Import` statement or if the business object is part of your solution. If you leave out <node>, the target node of the association is the root node of the business object. The following example shows the possible path expressions of an association in a complex business object:

```
import Common.DataTypes as CommonDTs;
```

```
businessobject BusinessObject1 {
  element Name : LANGUAGEINDEPENDENT_MEDIUM_Name;
  node Level1 {
    element Name : LANGUAGEINDEPENDENT_MEDIUM_Name;
    node Level2 {
      element Name : LANGUAGEINDEPENDENT_MEDIUM_Name;
      node Level3 {
        element Name : LANGUAGEINDEPENDENT_MEDIUM_Name;
      }
    }
  }
  association A_ToLevel3 to BusinessObject1.Level3;
  association B_ToLevel3 to Level3;
}
```

> **Note**
>
> If the target node is the root node, do not use the target node path in the association definition. If the target business object is the current business object, do not use the target business object in the association definition. However, this does not apply if the source **and** the target node is the root node of the current business object.

You can mark associations as relevant for access control by adding the annotation `RelevantForAccessControl`.

Examples

- **Association target within the current business object**
  `Item` is the name of the composition from the current node to the Item node.

```
association MainItem to Item;
```

  `Item` is the name of the composition from the current node to the Item node. `SubItem` is the name of the composition from the Item node to the `SubItem` node.

```
association ToSubItem to Item.SubItem;
```

- **Association to different business object**

```
association ToServReq to
AP.CRM.Global:ServiceRequest.ServiceRequestOverview using ID;
```

```
association ToBP to
AP.FO.BusinessPartner.Global:BusinessPartner using UUID;
```

- **Using keys and root node as a target node**
  The default key of target root node is used.

```
association Rule[0,1] to BonusRule;
```

  The key of target root node is specified.

```
association Material to Product using ProductID;
```

- **Target with fully qualified business object name**

```
association Customer [0,1] to
AP.FO.BusinessPartner.Global:Customer using ID;
```

- **Target with fully qualified business object name using an alias**

```
import AP.FO.BusinessPartner.Global as bp;
```

```
association Customer[0,1] to bp:Customer using ID;
```

- **Association marked as relevant for access control**

```
import AP.FO.BusinessPartner.Global as bp;
```

```
[RelevantForAccessControl] association ToBP to
bp:BusinessPartner;
```

See also

Alternative Key (Business Object) [page 194]

Valuation (Business Object) [page 215]

7.2.2.7 Business Object (Business Object)

Syntax	`businessobject <name> [ <Raises> ]? { [<Message>? <Node>] };`
Description	Use the *businessobject* keyword to specify the name and the nodes of a business object. A business object consists of a root node, which can contain child nodes. Even if not specified, the multiplicity of the root node is [1,1]. **i Note** The .bo file that contains the business object definition must have the same name as the business object itself. Each file contains the definition of a single business object.

Example	Business object, message and root node element. The root node raises a specified message:
	`businessobject BonusRule raises ReferenceInvalid {`
	`  message ReferenceInvalid text "Reference is invalid";`
	`  element ItemID : ID;`
	`}`

See also	Node (Business Object) [page 211]

7.2.2.8 Change History (Business Object)

Overview

Syntax	[ChangeHistory] `element` <element name> : <data type>;

Description	If you need to record the changes to a given value of your business object you use the **ChangeHistory** annotation. It will control the creation as well as the change of a value. It can be applied to the keywords: • `businessobject` It works in the same way as for any other node, but here it is applied to the root node of the business object only, not the whole business object. • `node` The changes of all elements (including the alternative keys used in associations) of this node are recorded. • `element` The changes of this element are recorded. If you provided the annotation already to the node of this element you will get an information message that this annotation is superfluous. It is not allowed to use this annotation together with the annotation `Transient`. • `association` The changes of this association are recorded. This is only supported for associations using an alternative key. For associations based on valuation please mark the elements used in the valuation with this annotation. **i Note** • The **ChangeHistory** annotation does not work for Dependent Objects. However, if the parent node is enabled for change history, the changes for Dependent Objects are tracked. • Annotating the keyword `businessobject` works only for the root node, not the complete business object.

Example	`businessobject` WorkingPlan `raises` DateMissing {
	
	`message` DateMissing `text` "Date is missing";
	
	[ChangeHistory] `element` ItemID : ID;
	
	`element` Name : LANGUAGEINDEPENDENT_MEDIUM_Name;
	
	[ChangeHistory] `node` Item [0,n] `raises` BonusRule.InvalidReference {
	
	`element` ItemID : ID;
	
	`element` ItemName : LANGUAGEINDEPENDENT_MEDIUM_Name;
	
	}
	
	`association` ToServReq `to` AP.CRM.Global:ServiceRequest.ServiceRequestOverview `using` ID;
	
	[ChangeHistory] `association` ToBP `to` AP.FO.BusinessPartner.Global:BusinessPartner `using` UUID;
	
	}

Related Information

Enable Change History [page 435]

7.2.2.9 Cross Deployment Unit (Business Object)

Syntax	`[CrossDeploymentUnit] association <name> to <path to target node> [using <composition path to target node>]?`
Description	You can use the `CrossDeploymentUnit` annotation to define an association from a node of a business object in one deployment unit to a node of a business object in another deployment unit. Optionally, you can specify the alternative key of the target node with the `using` keyword. **i Note** You cannot use a cross-deployment-unit association to access a business object in another deployment unit. You can only use a cross-deployment-unit association in an analytical data source to collect data from a business object residing in a different deployment unit.
Examples	`[CrossDeploymentUnit] association ToBusinessPartner to BusinessPartner;` `[CrossDeploymentUnit] association ToSalesOrder to SalesOrder using ID;`
See also	Association (Business Object) [page 196]

7.2.2.10 Description (Business Object)

Overview

Syntax	`[Description] node <name> { …`
Description	You can define the node as carrying language dependent descriptions. It will be represent as such in the Business Object model and can be used in the UI in a way that the description is automatically derived in the logon language. Ensure that at least one element or a supplementary component of an element of the node is of the data type `LanguageCode`

Example	`[Description] node ProductName [0,n] { element Name : LONG_Description; }`

7.2.2.11 Default Values (Business Object)

Syntax	Basic data types:
	`element <element name> : <data type name> = <default value>;`
	Structured data types:
	`element <element name> : <data type name> = {<component1> = <default value>, ..., <componentX> = <default value>};`
	You cannot use aggregated data types.

Description	Default values are initial values that are used to fill fields automatically when a new instance is created. The studio supports only static default values such as Boolean values, status information, numeric values, and free text. There is no syntax check for default values included in quotation marks. **i Note** Translation of default values is not supported.

Examples	Basic data types:
	`element Text1 : {content = "14", languageCode = "EN"};`
	`element Text2 : LANGUAGEINDEPENDENT_MEDIUM_Description = "Test Text";`
	`element Date : Date = "2012-11-01";`
	`element Time : Time = "23:59:59";`
	`element Integer : IntegerValue = 12345;`
	`element Code : ActionCode = "01";`
	Structured data type:
	`element Amount : Amount = {content = 13.45, currencyCode = "EUR"};`

See also	Data Types [page 92]

7.2.2.12 Dependent Object (Business Object)

Syntax	`[DependentObject(<dependent object name>)] [MultipleTexts]? node <name>;`

Description

A dependent object is a reusable part of a business object that - from a business point of view - cannot stand alone, but can only be used in the context of the business object. You can include the `TextCollection` and `AttachmentFolder` dependent objects in a business object definition.

> **i Note**
>
> For the `TextCollection` dependent object you can use the following:
>
> - `MultipleTexts` annotation to support different text types
> - `LanguageIndependent` annotation to support language independent text

In order to define the dependent object inclusion node, you have to specify a name. You can either use the dependent object name itself, for example, `TextCollection`, or add a specific meaning to the dependent object name, for example, `ProcurementSpecificationTextCollection`. If you include a dependent object in a business object definition several times, you should provide different names for each of the dependent object inclusion nodes. You can, for example, use the following syntax: `<subordinate node name> + <dependent object name>`, such as `ItemAttachmentFolder`.

> **i Note**
>
> You can add the same dependent object only once to a business object node.

> **i Note**
>
> If you choose to add the reuse UI, you need to provide a **Type Code** to determine the kind of `TextCollection` or `AttachmentFolder` you intend to use. The list of codes which are supported are as shown below:
>
> - `TextCollection`
> - 10001 Additional External Comment
> - 10002 Body Text
> - 10005 Rejection Reasons
> - 10006 Detailed Description
> - 10011 Internal Comment
> - 10025 Reason
> - 10026 Purchasing Note
> - 10027 Sales Note
> - 10029 Approval Note
> - 10047 Error Symptom
> - 10071 Work Description
> - `AttachmentFolder`
> - 10001 Standard Attachment
> - 10006 Certificate
> - 10013 Image
> - 10047 Multimedia
> - 10049 Maps
> - 10051 Webpage

Example

```
[DependentObject(TextCollection)] node ItemTextCollection;
```

Example with two annotations

```
[DependentObject(TextCollection)] [MultipleTexts] node
TextCollection;
```

7.2.2.13 Deployment Unit (Business Object)

Syntax

Assignment to a different deployment unit:

```
[DeploymentUnit(<deployment unit name>)] businessobject <name>
[ <Raises> ]? { [<message>? <node>] };
```

Cross-deployment-unit association:

```
[CrossDeploymentUnit] association <name> to <composition path to
target node> [using <alternative key>]?;
```

Description	A deployment unit is a piece of software that can be operated on a separate physical system, isolated from other pieces of software. The SAP cloud solution is organized in deployment units that group semantically related business objects, for example, Customer Relationship Management. There is also a special Foundation deployment unit that mainly contains master data objects, for example, Material, Employee, and Customer. Each business object resides in a deployment unit. You can explore SAP business objects that are released with the public solution model (PSM) and the deployment units in the *Repository Explorer* tool window. For more information, see Repository Explorer [page 49]. When a solution is created, the solution has to be assigned to a deployment unit. The business objects that are created subsequently in the solution are assigned to this default deployment unit. If you want to assign a business object to a different deployment unit, use the annotation `DeploymentUnit` at the beginning of the business object definition. **i Note** You cannot create your own deployment units. You can directly access business objects that reside in the Foundation deployment unit from any deployment unit. However, business objects in the Foundation deployment unit cannot access any business object residing in another deployment unit, not even by using asynchronous communication. You can only use a query to read the data of this business object. For all other deployment units the following applies: You cannot directly change the data of a business object residing in another deployment unit. However, you can read the data of such a business object by using a query, the `Retrieve` method or an association. To change the data of a business object residing in another deployment unit, use asynchronous, message-based communication. For more information, see Configure Internal Communication [page 579]. **i Note** You cannot use a cross-deployment-unit association to access a business object in another deployment unit. You can only use a cross-deployment-unit association in an analytical data source to collect data from a business object residing in a different deployment unit.

Example	Assigning a business object to a different deployment unit:
	
	``
	[DeploymentUnit(CustomerRelationshipManagement)] businessobject
	BonusRule {
	element ID : ID;
	}
	``

See also	Association (Business Object) [page 196]

7.2.2.14 Element (Business Object)

Syntax	`[ <annotations> ]? element <name> : <data type name>;`
Description	Use the *element* keyword to define an element of a business object node. The element name must be unique within the node. The element type can be a simple data type, for example, `ID`, or it can be a structured data type such as `Amount`. You can import data types from different namespaces. However, you cannot define new data types. For more information on importing entities, see Import (Business Object) [page 208]. **i Note** When defining an identifier element, we recommend you use the `ID` data type instead of the `Identifier` data type. The `ID` data type always has 60 characters, is not structured and does not contain the `Content` element. The data type supports, for example, upper case conversion and alpha conversion. Alpha conversion fills purely numeric user input like "4711" with leading zeros from the left side to allow better sorting in character fields. For example, if alpha conversion is not used, an alphabetic sorting would lead to results such as: "1", "10", "100", "1000", "2", "20", "200", and so on. You can define additional properties of an element by using annotations. For example, the annotation `AlternativeKey` defines the key element of a business object node. Key elements uniquely identify a node and can be used to identify the target of an association. An alternative key, `SAP_UUID` of type `UUID`, is generated implicitly for each node. Therefore, you do not need to define an alternative key for each node. Enclose annotations in square brackets.

Examples	Element with simple type:
	`element Count : IntegerValue;`
	Element with complex type:
	`element Target : Amount;`
	Key element:
	`[AlternativeKey] element EmployeeID : ID;`
	Element with fully qualified type name:
	`element Recipient : AP.Common.GDT:Date;`

See also	Association (Business Object) [page 196]

7.2.2.15 Extension (Business Object)

Overview

Syntax	`[Extension] businessobject <name> {<Node>*};`
Description	You use the `Extension` annotation to extend a business object. The name of the business object to be extended must be fully qualified, that is, it must include the name-space as well as the name, for instance, `AP.CRM.Global:Lead`. You can also specify all the extendable nodes.

Example	`[Extension] businessobject AP.CRM.Global:SalesOrder {`
	`  node Item {`
	`  }`
	`};`

See also	Syntax for Business Object Extension Definitions [page 218]

7.2.2.16 Import (Business Object)

Syntax	`import <namespace name>;`
Description	Use the import keyword to import business objects and data types from namespaces that are available in the repository. The namespace name is typically a Uniform Resource Identifier (URI) in dot notation (see example below). You can use imported business objects and data types in the definition of associations, elements, and messages. You can reference the imported objects directly by their names instead of using fully qualified names. If you use a namespace only for one or two sub-entities, you can omit the import statement and use the qualified name `<namespace name>:<name>` instead. If there are no conflicts for any entity in the business object within all imported namespaces, you can omit the relevant namespace, as the system can identify it automatically. **i Note** If you use more than one namespace in your business object definition, we recommend that you define an alias for each imported namespace and qualify business objects or data types from these namespaces by using the aliases. You do not need to import business objects and data types from the namespace of the current project but you can access them directly by their names. You can use code completion for importing the namespace as well as for accessing imported objects, that is, when you select the target business object of an association or the data type of a node element or message parameter.

Examples	**Import of a namespace:**
	`import AP.FO.BusinessPartner.Global;`
	Importing a namespace and defining an alias:
	`import AP.Common.GDT as apc;`
	`import ESF as esfnsp;`
	Using the defined aliases:
	`element duration : apc:Duration;`
	`message ReferenceInvalid text "Reference & is invalid": apc:NumberValue;`
	`association Customer to esfnsp:BusinessPartner;`

7.2.2.17 Label (Business Object)

Syntax	`[Label ("<label name>")] element <name> : <data type>;`
Description	You can define a label for a business object field. If you then add the field to a screen, the label is displayed on the screen instead of the field name. You can translate the label text by exporting it to an XLFF file. For more information, see the Translation Quick Guide [page 651].
Example	`[Label ("My label for field 1")] element ROOT_Field1 : LANGUAGEINDEPENDENT_EXTENDED_Text;`

Related Information

Add a Label and Tooltip to a Business Object [page 419]

7.2.2.18 Message (Business Object)

Syntax	`message <message ID> text "<message text>" [: <data type 1>[, <data type 2>[, <data type 3>[, <data type 4>]?]?]?]? ;`
Description	Use the *message* keyword to define a message between two business objects. The definition of a message includes the message ID, the message text, and, optionally, data types of up to four message parameters. The parameter names are defined by the system. The limit for a message ID is 30 characters and for a message text is 73 characters. The message ID must be unique within the business object; use uppercase for letters. You can reference data types from different namespaces. For more information, see Element (Business Object) [page 207] and Import (Business Object) [page 208].

Example	**Message without parameters:**
	
	`message DateMissing text "Due date of invoice is missing";`
	
	Message with two parameters:
	
	`message ReferenceInvalid text "Reference to sales order &1 invalid for &2" : IntegerValue, ESF:AdressType;`

See also	Raises (Business Object) [page 212]

7.2.2.19 Language Independent (Business Object)

Overview

Syntax	`[DependentObject(<dependent object name>)] [LanguageIndependent]? node <name>;`
Description	You can use the `LanguageIndependent` annotation together with the `TextCollection` dependent object to support language independent text. If the `TextCollection` dependent object is assigned to the current node, the `LanguageIndependent` annotation defines that no language code must be provided together with the text. If the user logs on with a different language than the one used during the creation of the text, the same text will be retrieved.

Example	`[DependentObject(TextCollection)]` `[LanguageIndependent] node TextCollection;`

Related Information

Dependent Object (Business Object) [page 203]

7.2.2.20 Multiple Texts (Business Object)

Syntax	`[DependentObject(<dependent object name>)] [MultipleTexts]? node <name>;`
Description	You can use the `MultipleTexts` annotation together with the `TextCollection` dependent object to support different text types. If the `TextCollection` dependent object is assigned to the current node, the `MultipleTexts` annotation defines if more than one text is allowed.
Example	`[DependentObject(TextCollection)] [MultipleTexts] node TextCollection;`

Related Information

Dependent Object (Business Object) [page 203]

7.2.2.21 Node (Business Object)

Syntax	`node <name> [ <multiplicity> ]? [ <Raises> ]? {[ <Element>, <Action>, <Association>, <Node> ]* };`

Description	Use the *node* keyword to define a business object node and its features, that is, elements, actions, node associations, child nodes, and raised messages. The name of the node must be unique within the business object. For the business object, the root node of the node hierarchy is automatically created. The multiplicity of the root node is [1,1]. Each direct or indirect child node implicitly contains a `ToRoot` association. A node must contain at least one explicitly defined element (not including `SAP_UUID`, which is implicitly generated for each node). Elements, actions, associations, and child nodes must have unique names within the node. The definition of a child node implicitly introduces a composition, that is, a relationship between a parent node and a child node. The multiplicity of the child node defines the upper and lower bound of that composition, that is, how many instances of the child node can be part of the parent node. The multiplicity of a node can be [0,1], [1,1], [0,n], or [1,n]. If no multiplicity is defined, the multiplicity [0,1] is used by default. You can use the composition to access features of the child node from the parent node (for example, in action implementations). For each composition, the reverse `ToParent` association is created implicitly. `SelectAll` and `ByElements` queries are created automatically for each node. It is not possible to define new queries in the business object.

Example	Business object and *raises* statement for the root node, message definition and root node element as well as subnode with a *raises* statement followed by a subnode element:
	
	`businessobject WorkingPlan raises DateMissing {`
	`  message DateMissing text "Date is missing";`
	`  element ItemID : ID;`
	`  node Item [0,n] raises BonusRule.InvalidReference {`
	`  element ItemID : ID;`
	`  }`
	`}`

See also	Association (Business Object) [page 196]

7.2.2.22 Raises (Business Object)

Syntax	`raises <message ID> [, <message ID>[,...]?]? ;`
Description	At least one message ID must follow the *raises* statement. Following the message ID, you can enter as many messages as needed. If the message is not part of the same business object, you must also specify the message group. The name of the message group is identical to the corresponding name of the business object.

Example

- *Raises* statement that raises a message from the same business object:

```
businessobject BonusRuleOne raises DateMissing {
  message DateMissing text "Date is missing";
  element ItemID : ID;
}
```

- *Raises* statement that raises messages from the same business object and messages with different message groups

```
businessobject BonusRuleTwo raises BonusRuleOne.DateMissing {
  message ReferenceInvalid text "Reference is invalid";
  element ItemID : ID;
  node Item raises BonusRuleOne.DateMissing, ReferenceInvalid {
    element ItemID : ID;
    }
  action Count raises ReferenceInvalid;
}
```

See also

Message (Business Object) [page 210]

Node (Business Object) [page 211]

7.2.2.23 Relevant for Access Control (Business Object)

Syntax

`[RelevantForAccessControl[(<parameter>)]?]`

Description

To do this, you mark associations [page 196] as relevant for access control by adding the annotation `RelevantForAccessControl`. The following conditions must be met:

- For each business object, you can only use one association that is relevant for access control.
- The source node of the association must be the root node.
- The target node of the association must be root node of the target business object.
- The target business object must have an association that is relevant for access control. The target of this association must be the dependent object `Access Control List`.
- The target business object must support at least one access context. You can find this information in the *Repository Explorer* by viewing the documentation of the relevant business object and by searching for the **Supported Access Context** section. If this section is not listed in the documentation, access context is not supported for this business object.

Example

```
[RelevantForAccessControl] association toBusinessPartner to
BusinessPartner;
```

See also

Define Access Control [page 424]

7.2.2.24 Separately Lockable (Business Object)

Overview

Syntax	[SeparatelyLockable] `node` <name> { <elements> };
Description	You can use this annotation to make sub-nodes separately lockable. This allows you to edit a node while another user edits another node of the same business object.
Example	[SeparatelyLockable] `node` Item { element id : ID };

7.2.2.25 Tooltip (Business Object)

Syntax	`[Tooltip ("<tooltip name>")] element <name> : <data type>;`
Description	You can define a tooltip for a business object field. If you then add the field to a screen, the label is displayed on the screen instead of the field name. You can translate the tooltip text by exporting it to an XLFF file. For more information, see the Translation Quick Guide [page 651].
Example	`[Tooltip ("My tooltip for field 1")] element ROOT_Field1 : LANGUAGEINDEPENDENT_EXTENDED_Text;`

Related Information

Add a Label and Tooltip to a Business Object [page 419]

7.2.2.26 Transient (Business Object)

Syntax	`[Transient] element <name> : <data type>;`

Description	You can use the `Transient` annotation to create transient fields. The data in such a field is not persistent, this means, it is not saved in the database. A transient field is filled after an `AfterLoading` event has been executed. The Transient annotation can be combined with the following annotations: • Label [page 209] • Tooltip [page 214]

Example	`[Transient] element myTransientField : LANGUAGEINDEPENDENT_LONG_Name;`
	
	`[Transient] [Label("MyTransientField")] element myTransientField : LANGUAGEINDEPENDENT_LONG_Name;`

7.2.2.27 Valuation (Business Object)

Syntax

```
association <name> [<multiplicity>] to <target node> valuation
(condition [&& condition]*);
<condition> ::= <element of target node> <operator> <element
defined in source node> | <code value>
<operator> ::= == | < | > | >= | <=
```

Description

The `valuation` keyword enables you to create not only 0,1 and 1,1 associations but also 0,n and 1,n associations. Thus, you can model hierarchical business object structures, for example, a product catalog.

You can add the `valuation` keyword to an association in order to evaluate the elements of a target node. A `valuation` association must have the multiplicity [0,1], [1,1], [0,n], or [1,n].

Usually, associations represent relations between nodes of the same business object or between nodes of different business objects. However, when you use the `valuation` keyword, the association must refer to a node within the same business object. Depending on the node type, the valuation is executed as follows:

- If the association refers to a **subnode** of a business object, the valuation is executed on the subnode instances that belong to the same business object instance as the source node instance (association within a business object instance).
- If the association refers to the **root node** of a business object, the valuation is executed on all instances of this business object (association across all instances of a business object).

The `valuation` keyword is followed by one or several conditions, which are evaluated when the associations are resolved. You define a condition as follows:

1. You specify the target element.
   This element must exist in the target node of the association.
2. You specify one of the following comparison operators:
   - == (is equal to)
   - < (is less than)
   - > (is greater than)
   - >= (is greater than or equal to)
   - <= (is less than or equal to)
   - ! (not equal)

   > **i Note**
   >
   > If you want to compare two node elements, you can only use the == operator.
3. You define either an element defined in the source node or a code value.

You can use the AND operator (`&&`) to connect several conditions. Please note that in this case the OR operator (`||`) and the NOT operator (`!`) are **not** supported.

In a condition, you can specify elements with non-structured data types only. However, you can use the element path, for example, `Amount.content`, to use elements of structured data types.

> **i Note**
>
> The `valuation` keyword cannot be used together with the `using` keyword and the `RelevantForAccessControl` annotation.

Examples

Valuation within a business object instance:

```
businessobject BusinessObject1 {
  [AlternativeKey] element ID:ID;
  node ChildHierNode[0,n] {
    element ID:ID;
    element ParentID:ID;
    association toChild [0,n] to ChildHierNode valuation (ParentID
== ID);
  }
}
```

In the real world, `BusinessObject1` may represent a catalog, containing sub-items that are ordered in a hierarchy (all items belong to the same catalog)

Valuation across all instances of a business object:

```
businessobject BusinessObject1 {
  [AlternativeKey] element ID:ID;
    element ParentID:ID;
    association toChild [0,n] to BusinessObject1 valuation
(ParentID == ID);
}
```

In the real world, `BusinessObject1` may represent persons or objects (for example, buildings) that build a hierarchy.

Valuation with a code value:

```
businessobject BusinessObject1 {
  [AlternativeKey] element ID:ID;
  node CountryNode [0,n]{
    element ID:ID;
    element CountryCode:CountryCode;
  }
  association toGermany [0,1]to CountryNode valuation (CountryCode
== "DE");
}
```

See also

Association (Business Object) [page 196]

Relevant for Access Control (Business Object) [page 213]

7.2.2.28 With Administrative Data (Business Object)

Overview

Syntax

[WithAdministrativeData] `businessobject` <business object name> {

[WithAdministrativeData] `node` <node name> {

Description	If you want to record the creation date and time, along with the identity of the creator and/or the user who made the last changes, you use the `WithAdministrativeData` annotation. This annotation only supports the Business Object definition and the Node definition.
Example	[WithAdministrativeData] `businessobject` Businessbject1 { [WithAdministrativeData] `node` Item [0,n] {

7.2.3 Business Object Extension Definitions

7.2.3.1 Syntax for Business Object Extension Definitions

Use the keywords and functions listed below to describe the structure and interface of a business object extension in the studio.

For more information about the general syntax conventions and rules, see Scripting Language [page 189].

Naming Rules in Business Object Extensions

Comply with the following naming rules:

- The name of an entity must start with a letter and can contain letters, digits, and underscore characters only. Do not start a name with “SAP”.
- Entities within a business object cannot have the same name as the business object.
- Entities cannot use names that already exist in the business object being extended.
- Do not use the following technical terms as an entity name:
  - association
  - businessobject
  - element
  - import
  - namespace
  - node
  - query
  - to
  - type
  - using

Extension Template

When you create a business object extension, you are presented with a template that includes the extendable nodes.

Taking the Sales Order business object as an example, the extension template begins with the annotation `[Extension]` and contains the extendable node `Item`. The `Root` node is also extendable, although not explicitly shown.

```
import AP.Common.GDT;
import AP.CRM.Global;

[Extension] businessobject AP.CRM.Global:Lead {
// You must activate this business object before you can access the extension
fields
// in script files and screens

node Item {
}
}
```

Code Completion

Based on the code you type, code completion presents you with a list of valid options. For example, when you type a colon (`:`) after an element name, you will see a list of valid data types. If there is only one valid option, the code is completed at once.

You can also use **CTRL** + **SPACEBAR** to force the code completion function to display the valid options at the current cursor location.

Comments

You can enter comments using the following syntax:

- **comment line**

  ```
  // <comment text>
  ```

  A comment line starts with two slashes (//) and spans to the end of the current line.

- **comment block**

  ```
  /* <comment text>
  <comment text>
  <comment text>
  */
  ```

  A comment block starts with a forward slash followed by an asterisk and ends with an asterisk followed by a forward slash.

List of Keywords

- Action (Business Object Extension) [page 220]

- Business Object (Business Object Extension) [page 220]
- Element (Business Object Extension) [page 224]
- Import (Business Object Extension) [page 225]
- Message (Business Object Extension) [page 226]
- Node (Business Object Extension) [page 227]
- Raises (Business Object Extension) [page 231]

List of Annotations

- Decimal (Business Object Extension) [page 222]
- Label (Business Object Extension) [page 226]
- Reference (Business Object Extension) [page 232]
- Relation (Business Object Extension) [page 233]
- Scenario (Business Object Extension) [page 233]
- Tooltip (Business Object Extension) [page 234]

7.2.3.2 Action (Business Object Extension)

Syntax	`action <name> [<Raises>]? ;`
Description	Use the *action* keyword to define an action for a node. Actions carry out the business logic and are implemented in .absl files. You can only use this keyword in business object extensions created on the ESF2 framework.

Example	Definition of an action:
	`action Cancel;`
	Action definition with *raises* statement :
	`action CountHappyDays raises BonusRule.ReferenceInvalid, DateMissing;`

See also	Raises (Business Object) [page 212]

7.2.3.3 Business Object (Business Object Extension)

Syntax	`[Extension] businessobject <name> { <Node>* };`

Description	The *businessobject* keyword specifies the business object being extended.

Example	`[Extension] businessobject AP.CRM.Global:Lead {`
	`  node Item {`
	`  }`
	`}`

7.2.3.4 Annotation (Business Object Extension)

Syntax	`[<annotation>[(<parameter>)]?]`
Description	Annotations are used to add a category or property to an object specified by the subsequent keyword. The names of the annotations are predefined and can be displayed by the code completion. Technically, you can combine as many annotations as you want. But not all combinations are possible and not all of them make sense. For more information, see the list of possible combinations below.
Example	`[annotation]? element name : datatype;`

Annotation Overview

Annotation	Argument	Keyword	Can Be Combined With	Allowed in Standard/ Extension Node
AlternativeKey (Extension Node) [page 194]	—	element	Label, Tooltip	Extension
ChangeHistory (Extension Node) [page 199]	—	node, element, Association	—	Extension
Decimal (Business Object Extension) [page 222]	digits, fraction	element	Label, Scenario, Tooltip, Transient	Standard
Description (Extension Node) [page 201]	—	node	SeparatelyLockable	Extension
Label (Business Object Extension) [page 226]	content	element	Decimal, Relation, Scenario, Tooltip, Transient	Both
Label (Extension Node) [page 209]	content	element	AlternativeKey, Tooltip, Transient	Both

Reference (Business Object Extension) [page 232]	—	element	Scenario	Standard
Relation (Business Object Extension) [page 233]	list of names	element	Label, Scenario, Tooltip	Standard
Scenario (Business Object Extension) [page 233]	list of names	element	Decimal, Label, Reference, Relation, Tooltip	Standard
SeparatelyLockable (Extension Node) [page 214]	—	node	Description	Extension
Tooltip (Business Object Extension) [page 234]	content	element	Decimal, Label, Relation, Scenario, Transient	Both
Tooltip (Extension Node) [page 214]	content	element	AlternativeKey, Label, Transient	Both
WithAdministrativeData (Extension Node) [page 217]	—	node, businessobject	—	Extension

7.2.3.5 Decimal (Business Object Extension)

Syntax	`[Decimal (x,y)] element <name> : DecimalValue;`
Description	You can declare a decimal value with a specific number of digits before and after the decimal point. The decimal annotation [Decimal (x,y)] requires two parameters: • x = the total number of digits allowed, up to a maximum of 29 • y = the number of digits allowed after the decimal point, up to a maximum of 14 The first parameter value must be greater than the second parameter value. **i Note** • The default number of digits before and after the decimal point are 7 and 2 respectively. • When you use negative numbers, the minus sign is counted as a digit. For example, `[Decimal (2,0)]` allows 55 and -5 but not -55.

Example	Valid Syntax
	`[Decimal (2,1)] element Test_Decimal_01 : DecimalValue;`
	– possible numbers: 5, 55, -5, 5.5, 0.5
	`[Decimal (6,2)] element Test_Decimal_02 : DecimalValue;`
	– possible numbers: 5555.55, -555.55, 555555, 0.55
	Invalid Syntax
	`[Decimal (4,4)] element Test_Decimal_03 : DecimalValue;`
	`[Decimal (6,8)] element Test_Decimal_04 : DecimalValue;`

7.2.3.6 Default Values (Business Object Extension)

Syntax	Basic data types:
	`element <element name> : <data type name> = <default value>;`
	Structured data types:
	`element <element name> : <data type name> = {<component1> = <default value>, <componentX> = <default value>};`
	You cannot use aggregated data types.

Description	Default values are initial values that are used to fill fields automatically when a new instance is created. The studio supports only static default values such as Boolean values, status information, numeric values, and free text. There is no syntax check for default values included in quotation marks. **i Note** Translation of default values is not supported.

Examples

Basic data types:

```
element Text1 : {content = "14", languageCode = "EN"};
```

```
element Text2 : LANGUAGEINDEPENDENT_MEDIUM_Description = "Test
Text";
```

```
element Date : Date = "2012-11-01";
```

```
element Time : Time = "23:59:59";
```

```
element Integer : IntegerValue = 12345;
```

```
element Code : ActionCode = "01";
```

```
[Decimal(12.5)] element MyOKDecimal2 : DecimalValue = 1234.12345
```

Structured data type:

```
element myOKAmount1 : Amount = {content = 13.45, currencyCode =
"EUR"};
```

See also

Data Types [page 92]

7.2.3.7 Element (Business Object Extension)

Syntax

```
[ <Annotation> ]? element <name> : <data type>;
```

Description

Use the *element* keyword to define an extension element for a business object node.

The element name must be unique within the solution and cannot exceed 60 characters.

The element type can be a simple data type, for example, `LANGUAGEINDEPENDENT_EXTENDED_Text`, or it can be a structured data type such as `Amount`. You can import data types from different namespaces. However, you cannot define new data types. For more information on importing entities, see Import (Business Object Extension) [page 225].

Data Types	The following data types are available by default: • Amount • Date • DecimalValue **i Note** The default number of digits before and after the decimal point are 7 and 2 respectively. • CodeList – SAP code lists • ID • Identifier – SAP identifiers • Indicator • Quantity • Time • Text **i Note** If you use this data type in a business object extension, the length of the corresponding field is **not** restricted to a specific number of characters. Only in business object definitions, the field length is restricted to 255 characters. • WebURI • EmailURI By importing additional namespaces, you can access additional SAP CodeLists and Identifiers. You can also use CodeLists that you have defined within your solution.
Example	`element Target : Amount;`

7.2.3.8 Import (Business Object Extension)

Syntax	`import <namespace name>;`
Description	Use the *import* keyword to import data types from namespaces that are available in the repository. The namespace name is typically a Uniform Resource Identifier (URI) in dot notation (see example below). You can use imported data types in the definition of elements. You can reference the imported data types directly by their names instead of using fully qualified names. If you use a namespace only for one or two sub-entities, you can omit the *import* statement and use the qualified name `<namespace name>:<name>` instead. If there are no conflicts for any entity in the business object within all imported namespaces, you can omit the relevant namespace, as the system can identify it automatically. You can also define an alias for an imported namespace and qualify data types from this namespace by using the alias.

Example	Import of a namespace:
	`import AP.Common.GDT;`
	Definition of an alias:
	`import AP.Common.GDT as APC;`
	Using the defined alias:
	`element Target : APC:Amount;`

7.2.3.9 Label (Business Object Extension)

Syntax	`[Label ("<label name>")] element <name> : <data type>;`
Description	You can define a label for an extension field. If you then add the extension field to a screen, the label is displayed on the screen instead of the extension field name. **i Note** If the element is structured, such as `Amount`, the label annotation is defined for the `.content` element. Please note that you **cannot** define a label for a structured element that does not contain a `.content` element or that contains more than one `.content` element.
Example	`[Label ("Screen Label")] element Opportunity_BO_extension_field : Indicator;`

7.2.3.10 Message (Business Object Extension)

Syntax	`message <message ID> text "<message text>" [: <data type 1>[, <data type 2>[, <data type 3>[, <data type 4>]?]?]?]?;`
Description	Use the *message* keyword to define a message for a business object extension. The definition of a message includes the message ID, the message text which can include up to four parameters, and the data types of any parameters used. The limit for a message ID is 30 characters and for a message text is 73 characters. The message ID must be unique within the business object extension. Messages can only be raised in the business object extension where they are defined. In addition, messages must be assigned to the individual nodes where they are to be called.

Example

```
import AP.Common.GDT;
import AP.CRM.Global;
[Extension] businessobject AP.CRM.Global:Opportunity raises
Information_Message, Error_Message, Warning_Message,
Success_Message {
// You must activate this business object before you can access the
extension fields
// in script files and screens.
message Information_Message text "This is an Information message";
message Error_Message text "This is an Error message with one
parameter &1" : LANGUAGEINDEPENDENT_SHORT_Name;
message Warning_Message text "This is a Warning message with two
parameters &1 and &2" : LANGUAGEINDEPENDENT_SHORT_Name,
IntegerValue;
message Success_Message text "This is a Success message with a
variable parameter &1" : Amount;
element Test_Extension_Field : Amount;
  node Item raises Information_Message {
  }
}
```

7.2.3.11 Node (Business Object Extension)

Syntax

```
node <name> [ <multiplicity> ]? {[ <Element>, <Node> ]* };
```

Description

The *node* keyword specifies the extendable nodes within a business object.

You can create and add your own custom nodes within a business object extension. However, you must note that not all standard business objects support these node extensions. For more information on the list of standard business objects, see below..

Example

```
[Extension] businessobject AP.CRM.Global:SalesOrder {
  node Item {
  }
}
```

The following business objects support node extensions:

- In **SAP Business ByDesign**,

Business Object Name	Namespace
AccountingCoding-BlockDistribution	http://sap.com/xi/AP/IS/CodingBlock/Global
AddressSnapshot	http://sap.com/xi/AP/FO/Address/Global
BankDirectoryEntry-File	http://sap.com/xi/AP/FO/BankDirectoryEntry/Global

Business Object Name	Namespace
BusinessPartnerHierarchy	http://sap.com/xi/AP/FO/BusinessPartner/Global
CodingBlockCustomObject1	http://sap.com/xi/AP/IS/CodingBlock/Global
CompanyFinancialsProcessControl	http://sap.com/xi/AP/FO/CompanyFinancialsProcessControl/Global
EmployeeTimeCalendar	http://sap.com/xi/AP/TimeAndLabourManagement/Global
FinancialAccountingViewOfPurchasingDocument	http://sap.com/xi/AP/FinancialAccounting/Global
FinancialPlanData	http://sap.com/xi/AP/FinancialAccounting/Global
FinancialsViewOfContract	http://sap.com/xi/AP/FinancialAccounting/Global
ForeignTradeProductClassification	http://sap.com/xi/FTA/ForeignTradeProductClassification/Global
IdentityBusinessRole	http://sap.com/xi/AP/PC/IdentityManagement/Global
ObjectIdentifierMapping	http://sap.com/xi/CommunicationServicesManagement/Global
Payment	http://sap.com/xi/AP/Payment/Global
PaymentBatch	http://sap.com/xi/AP/Payment/Global
ProjectStockOrder	http://sap.com/xi/AP/ProjectManagement/Global

- In **SAP Cloud for Customer**,

Business Object Name	Namespace
AccountingCodingBlockDistribution	http://sap.com/xi/AP/IS/CodingBlock/Global
Activity	http://sap.com/xi/AP/PC/ActivityManagement/Global
ActivityWorklist	http://sap.com/xi/AP/FO/Activity/Global
AddressSnapshot	http://sap.com/xi/AP/FO/Address/Global
AttachmentFolder	http://sap.com/xi/AP/FO/Address/Global

Business Object Name	Namespace
BusinessActivityIndustryExtension	http://sap.com/xi/AP/PC/ActivityManagement/Global
BusinessAttribute	http://sap.com/xi/AP/FO/BusinessAttribute/Global
BusinessAttributeAssignment	http://sap.com/xi/AP/FO/BusinessAttribute/Global
BusinessAttributeSet	http://sap.com/xi/AP/FO/BusinessAttribute/Global
BusinessPartnerRelationship	http://sap.com/xi/AP/FO/BusinessPartnerRelationship/Global
BusinessPartnerSolutionExtensions	http://sap.com/xi/AP/FO/BusinessPartner/Global
BusinessPartnerVisitingInformation	http://sap.com/xi/AP/FO/BusinessPartner/Global
Campaign	http://sap.com/xi/AP/CRM/Global
CampaignResponseOption	http://sap.com/xi/AP/CRM/Global
CashDiscountTerms	http://sap.com/xi/AP/FO/CashDiscountTerms/Global
CodTimeEntry	http://sap.com/xi/AP/CRM/Global
CodTimeReport	http://sap.com/xi/AP/CRM/Global
ComplaintManagementSalesOrder	http://sap.com/xi/AP/CRM/Global
Contract	http://sap.com/xi/AP/CRM/Global
CustomerQuote	http://sap.com/xi/AP/CRM/Global
DesignRegistration	http://sap.com/xi/AP/CRM/Global
ExternalAggregatedCustomerSalesOrderView	http://sap.com/xi/AP/FO/CustomerDocumentReplication/Global
ExternalCustomerKPIView	http://sap.com/xi/AP/FO/CustomerDocumentReplication/Global
ExternalIdentity	http://sap.com/xi/AP/FO/BusinessPartner/Global
ExternalPriceDocument	http://sap.com/xi/AP/FO/PriceAndTax/Global
IdentityBusinessRole	http://sap.com/xi/AP/PC/IdentityManagement/Global
InstallationPoint	http://sap.com/xi/AP/FO/IBase/Global
InstalledBase	http://sap.com/xi/AP/FO/IBase/Global
InsurancePolicy	http://sap.com/xi/AP/CRM/Global

Business Object Name	Namespace
InsuranceQuickQuote	http://sap.com/xi/AP/CRM/Global
InsuranceQuote	http://sap.com/xi/AP/CRM/Global
InsuranceRenumerationAnalytics	http://sap.com/xi/AP/CRM/Global
Lead	http://sap.com/xi/AP/CRM/Global
LeadIndustryExtn	http://sap.com/xi/AP/CRM/Global
MaintenancePlan	http://sap.com/xi/AP/CRM/Global
MarketingPermission	http://sap.com/xi/AP/CRM/Global
ObjectIdentifierMapping	http://sap.com/xi/CommunicationServicesManagement/Global
OperatingHours	http://sap.com/xi/AP/FO/OperatingHours/Global
Opportunity	http://sap.com/xi/AP/CRM/Global
OpportunityIndustryExtension	http://sap.com/xi/AP/CRM/Global
OrganisationAddress	http://sap.com/xi/AP/FO/Address/Global
PartnerAddress	http://sap.com/xi/AP/FO/Address/Global
PdmMaterialIndustryExtn	http://sap.com/xi/AP/FO/ProductDataMaintenance/Global
ProductBusinessPartnerRelation	http://sap.com/xi/AP/FO/ProductBusinessPartnerRelationManagement/Global
ProfessionalServicesMasterProject	http://sap.com/xi/AP/CRM/Global
ProjectAlert	http://sap.com/xi/AP/CRM/Global
Promotion	http://sap.com/xi/AP/CRM/Global
Questionnaire	http://sap.com/xi/AP/FO/Activity/Global
Route	http://sap.com/xi/AP/PC/ActivityManagement/Global
SalesTerritory	http://sap.com/xi/AP/PC/SalesTerritoryMgmt/Global
SalesTerritoryIndustryExt	http://sap.com/xi/AP/PC/SalesTerritoryMgmt/Global
ServiceIssueCategoryCatalogue	http://sap.com/xi/AP/FO/ServiceIssue/Global
ServiceRequest	http://sap.com/xi/AP/CRM/Global

Business Object Name	Namespace
SupplierInvoiceRequest	http://sap.com/xi/AP/PC/SupplierInvoiceProcessing/Global
TargetGroup	http://sap.com/xi/AP/CRM/Global
TextCollection	http://sap.com/xi/DocumentServices/Global
UtilitiesBusinessAgreement	http://sap.com/xi/AP/CRM/Global
UtilitiesConnectionObject	http://sap.com/xi/AP/CRM/Global
UtilitiesPointOfDelivery	http://sap.com/xi/AP/CRM/Global
VisitTourPlan	http://sap.com/xi/AP/PC/ActivityManagement/Global
WorkplaceAddress	http://sap.com/xi/AP/FO/Address/Global

7.2.3.12 Raises (Business Object Extension)

Syntax	`raises <message ID> [, <message ID>[,...]?]?;`
Description	Use the *raises* keyword to declare messages for a business object extension. At least one message ID must follow the *raises* statement. Messages can only be raised in the business object extension where they are defined. In addition, messages must be assigned to the individual nodes where they are to be called.
Example	

```
import AP.Common.GDT;
import AP.CRM.Global;
[Extension] businessobject AP.CRM.Global:Opportunity raises
Information_Message, Error_Message, Warning_Message,
Success_Message {
// You must activate this business object before you can access the
extension fields
// in script files and screens.
message Information_Message text "This is an Information message";
message Error_Message text "This is an Error message with one
parameter &1" : LANGUAGEINDEPENDENT_SHORT_Name;
message Warning_Message text "This is a Warning message with two
parameters &1 and &2" : LANGUAGEINDEPENDENT_SHORT_Name,
IntegerValue;
message Success_Message text "This is a Success message with a
variable parameter &1" : Amount;
element Test_Extension_Field : Amount;
  node Item raises Information_Message {
  }
}
```

7.2.3.13 Reference (Business Object Extension)

Overview

Syntax	`[Reference] element <extension_field_name>;`
Description	The `Reference` annotation enables you to use an element declared in one business object extension in another business object extension. You can also use an element in the item node that is declared in the root node of the same business object extension. The `Reference` annotation cannot be combined with the following annotations: • Decimal • Relation • Label • Tooltip • Transient **i Note** You do not need to include a data type when referencing an element, as the data type is specified in the original element definition.

Examples	1. **Reference from one business object extension to another business object extension**
	Definition of the first business object extension:
	`element Test_Extension_Field : Text;`
	Definition of the second business object extension:
	`[Reference] element Test_Extension_Field;`
	2. **Reference from an item node to the root node of the business object extension**
	Definition of the root node:
	`element Test_Extension_Field : Text;`
	Definition of the item node:
	`[Reference] element Test_Extension_Field;`

→ Tip

In contrast to the Relation [page 233] annotation, which refers to any business object element, the `Reference` annotation refers to a specific element within the same business object.

7.2.3.14 Relation (Business Object Extension)

Syntax	`[Relation(Namespace:BusinessObject.BO_Node.Element)] element <data type>;` with <data type with representation term `Identifier`>
Description	The `Relation` annotation enables you to create a reference to a business object by using the identifier of an element, for example, `ID`, `PartyID`, or `UUID` as the target of the reference. Thus, you can create an element that is like the element you are referring to and retrieve the data of this element, for example, for an object value selector or a query. Tooltips and labels are not retrieved from the element you are referring to. You need to specify the namespace in the annotation, if the namespace has not yet been specified in the definition of the business object extension. You can combine the `Relation` annotation with the `Scenario`, `Label`, and `Tooltip` annotations and with default values but not with the `Reference`, `Decimal`, and `Transient` annotations.

Examples	`[Relation(AP.CRM.Global:Opportunity.Item.ItemProduct.ProductKey.ProductID)] element MyProductID : ProductID;`
	
	`[Relation(AP.FO.BusinessPartner.Global:Employee.Identification.EmployeeID)] element MyEmployeeID : EmployeeID;`

7.2.3.15 Scenario (Business Object Extension)

Syntax	`[Scenario (<scenario_name>)] element <extension_field_name>:<data_type>;`

Description	A process extension scenario links the data from one business context to other related business contexts. You create an extension scenario from a list of predefined extension scenarios. For example, you can create an extension scenario that links the following business contexts:
	• *Lead - General Information > Opportunity - General Information*
	• *Opportunity - General Information > Sales - General Information*
	Each extension scenario contains one or more data flows. Each data flow consists of a source and target business context. This reflects the direction in which data is passed from one business context to the next as part of a business process.
	• *Opportunity - General Information > Sales - General Information*
	  ○ Opportunity - General Information > Sales Quote - General Information
	  ○ Opportunity - General Information > Sales Order - General Information
	i Note
	Extension scenarios are specific to the node for which they were created. If an extension scenario was created for the root node, then it can only be called at the root node within the `.xbo`. Similarly, an extension scenario created for the item node can only be called from the item node within the `.xbo`.
	If you now extend any of the business objects involved in the scenario, you can use the annotation `[Scenario]` to declare that the element is available to all the business contexts specified in the scenario.
	For example, if you add an element to the Lead business object, you can use an extension scenario to make this element available if you extend the Opportunity, Sales Quote, or Sales Order. When an Opportunity is created from the Lead, the data for your extension field is passed to the Opportunity. Similarly, when a Sales Order is created from the Opportunity, the data is passed to the Sales Order.

Example	`[Scenario (Sales_Lead_Tracking)] element Sales_Lead_Source : Text;`
See Also	For a list of all predefined extension scenarios, see Extension Scenarios [page 460].

7.2.3.16 Tooltip (Business Object Extension)

Syntax	`[Tooltip ("<tooltip name>")] element <name> : <data type>;`
Description	You can define a tooltip for an extension field. If you then add the extension field to a screen, the tooltip is displayed when you place the cursor on the extension field. **i Note** If the element is structured, such as `Amount`, the tooltip annotation is defined for the `.content` element. Please note that you **cannot** define a tooltip for a structured element that does not contain a `.content` element or that contains more than one `.content` element.

Example

```
[Tooltip ("Screen Tooltip")] element
Opportunity_BO_extension_field : Indicator;
```

7.2.4 Business Logic

7.2.4.1 Syntax for Implementation of Business Logic

The methods and keywords listed below enable you to implement the business logic for your solution in script files, such as actions, validations and events. You can implement, for example, `BeforeSave` events and `OnSave` validations, and the actions that you have defined for your business objects. For more information about the general syntax conventions and rules, see Scripting Language [page 189].

You can also use built-in functions and reuse libraries that extend the scope of the methods described below. For more information, see the following documentation:

- Built-In Functions [page 321]
- Basic Reuse Libraries [page 274]
- Reuse Libraries for Business Areas [page 308]

Type Handling

The scripting language supports type inference, thereby avoiding the need of explicitly identifying relevant types. This is especially useful for script files, where elements of existing business objects are accessed either by your own business objects or by SAP business objects. For structured element types, the code completion enables you to see the data type's structure while accessing the elements of the involved business objects.

For more information about the basic data types that the scripting language supports and their behavior, see Behavior of Data Types [page 329].

Code Completion

Apart from using a period (.) for the code completion of path elements, you can use the key combination **CTRL** + **J** to display all options for the current context. You can use this feature at any cursor location, for example, when the cursor is located at the end of a partially typed identifier or keyword or when the cursor is located within or directly before an already existing and valid identifier.

Method Tooltip

The method tooltip displays the documentation of a method and its parameter definitions. The method tooltip appears when you type an opening parenthesis, which is the method parameter indicator character.

To display the method tooltip, you can also use the key combination **CTRL** + **SHIFT** + **SPACEBAR** whenever the cursor is located within a method name or a parameter of a method call. If multiple methods with the same name but different parameters are available, you can choose the appropriate method signature from the list using the up and down arrow keys. If you use the left and right arrow keys, you can display the definitions of the individual parameters.

The method tooltip is displayed automatically when you type the parameters of a method. The method parameter description moves to the next parameter whenever you type a comma (,) before the closing parenthesis.

Comments

The syntax supports single-line and multiple-line (block) comments. A single-line comment starts with two slash marks (//) and spans to the end of the current line. The block comment, which can span multiple lines, starts with a slash mark and an asterisk (/*) and ends with an asterisk and a slash mark (*/).

Single-line comment:

```
// <comment>
```

Multi-line comment:

```
/* <comment> */
```

Code Formatter

The code formatter helps to maintain a clean and readable source code in script files. It offers the following formatting options:

- Properly indenting lines
- Inserting new lines, for example, before or after a control block
- Inserting spaces, for example, in expressions

When you have opened a script file in the code editor, you can execute the code formatter by selecting *Edit* > *Advanced Menu* > *Format Document*. Alternatively, you can press **CTRL** + **K** or **CTRL**+ **D**.

The code formatter formats the whole script file; you cannot use it to format only parts of the code.

Syntax

Simple and complex statements are supported. Complex statements usually contain a code body which is opened and closed by braces { }. The body, in turn, can contain any type of statement, that is, the statements can be nested. Simple statements, in contrast, can only be assignments and path expressions, in addition to variable declarations and the import statement.

List of Methods and Keywords

The methods and keywords are divided into the following sections:

- **Contexts**
  Contains information about the *this* keyword.
  - This [page 272]
- **Simple Statements**
  Contains information about statements such as the *import* keyword and the declaration of variables.
  - Import [page 253]
  - Declarations [page 247]
  - Assignment [page 240]
- **Complex Statements**
  Contains information about control and loop statements.
  - If Control Statement [page 253]
  - Switch Control Statement [page 272]
  - Foreach Loop Statement [page 249]
  - While Loop Statement [page 273]
  - Continue Statement and Break Statement [page 245]
- **Expressions**
  Contains information about all types of expressions, for example, literals, path expressions, and arithmetic expressions.
  - Literals [page 254]
  - Function Literals [page 251]
  - Logical Expressions [page 255]
  - Conditional Expressions [page 245]
  - Arithmetic Expressions [page 240]
  - Path Expressions [page 258]
  - Name Qualification [page 257]
  - Action Execution [page 239]
- **Association Handling**
  Contains information about accessing, setting, and checking associations.
  - Association Access [page 241]
  - Set Associations [page 270]
  - Check Associations (IsSet) [page 242]
  - Reset Associations [page 268]
- **Business Object Lifecyle**

Describes how the lifecycle of a business object instance can be controlled by the coding, that is, when creating, reading, updating and deleting a business object. For more information, see Business Object Lifecycle [page 242].

- Create Instance [page 246]
- Create with Reference Action [page 247]
- Query Execution [page 260]
- Retrieve Instance [page 268]
- Delete Instance [page 248]
- Check Node Existence(IsSet) [page 243]

- **Collection Handling**
  Contains information about how to define and modify collections. For more information, see Collection Handling [page 244].
  - Collection Definition [page 244]
  - Add to Collection [page 239]
  - Clear Collection [page 243]
  - Sort Collection (OrderBy) [page 271]
  - Filter Collection (Where) [page 249]
  - Remove from Collection [page 268]
  - Remove Duplicate Rows from Collection (DistinctBy) [page 267]
  - Get First or Get Last Row of Collection [page 252]
  - Count Rows of Collection [page 246]
- **Message Handling**
  Contains information about how to create and raise messages in the implementation phase.

  > **i Note**
  >
  > You cannot receive and handle messages that have been raised by a different script file containing implementation code or the business object infrastructure, for example, during the creation of a business object node instance.

  - Raise Message [page 267]
- **Service Integration Methods**
  Contains information about indicators and return values used for condition evaluation in service integrations scenarios. For more information, see Service Integration Methods [page 270].
  - Script File Parameter [page 270]
  - Return Values [page 269]
  - Process Context [page 259]

7.2.4.2 Action Execution (Business Logic)

Syntax	Static execution:
	`<business object name>[.<node name>].<action name>([<param>]*);`
	Instance-based execution:
	`<node instance var>.<action name>([<param>]*);`

Description	An action execution is a special form of a path expression. The code completion offers actions that are based on business object nodes in the same way as common method calls. You can enter parameters in parentheses (..). In reuse library method calls, you can also use optional parameters but not in actions that are based on business object nodes. Even if there are no parameters, you must always use parentheses. In contrast to reuse service methods, action methods based on business object nodes do not support return values. If the multiplicity of the action definition in the business object model is [0,0], you can execute the action statically by using static path expressions that identify a business object node. If the multiplicity is [1,1] or [1,n], you can execute the action instance-based. In this case, the instance identifier has to be a collection.

Example	• Instance-based execution ([1:1]):
	`import AP.CRM.Global;`
	`foreach (var so in SalesOrder.Header.Execute()) {`
	`  so.Item.CheckATP();`
	`  break;`
	`}`
	• Static execution ([0,0]):
	`BOName.NodeName.ActionName();`

7.2.4.3 Add to Collection (Business Logic)

Syntax	`<collection>.Add(<node instance or collection>);`
Description	Use the *Add* method to add a single row or an existing collection to the end of a collection. **i Note** This method cannot be applied to read-only collections. This is only relevant for code related to form data type extensions and in case of mass-enabled script files, in which the *this* keyword represents an immutable collection of nodes.

Example

```
var collection : collectionof Opportunity;
var dataOpportunity : elementsof Opportunity;
var nodeRef;
dataOpportunity.Name = "New opportunity 4711"
nodeRef = this.Create(dataOpportunity);
collection.Add(nodeRef);
```

7.2.4.4 Arithmetic Expressions (Business Logic)

Syntax

```
literal | <varName> | <path expression>

[ + | – | * | / | % ] <arithmetic expression>;
```

Description

The arithmetic expressions support the common mathematical operators for addition, subtraction, multiplication, division, and modulo calculations. Operands can be atomic expressions, such as literals, variable identifiers, path expressions, or other arithmetic expressions.

You can overwrite the precedence order of the operators by using parentheses.

The operands have to evaluate to the exactly same type. The compiler does not implicitly convert incompatible types. The plus sign (+) operator is overloaded to allow string concatenation.

Example

```
var result = (-0.234e-5 + this.Total) * currentValue;
var resultString = "Hello " + "world!";
```

7.2.4.5 Assignment (Business Logic)

Syntax

```
<x> = <y>;
```

Description

An assignment assigns the value of the expression on the right to the expression on the left. The expression on the left must be a variable or an element of a business object instance whereas the expression on the right can be any kind of expression. For the expressions on the right you can use static path expressions, for example, if you want to keep a copy of a reference to static query metadata. The default assignment uses copy semantics.

> **i Note**
>
> You cannot use chained assignments, that is, a = b = c.

You can also assign node instances on the right to associations on the left. The node instances on the right can either be a newly created node instance or a node instance that has been retrieved through a query. However, this kind of assignment is only possible for modeled associations with [1,1] multiplicity. Node instances to [1,n] associations, which are implemented and not modeled, are assigned implicitly when new node instances are created through create-enabled associations.

Example	`name = this.Name;`

7.2.4.6 Association Access (Business Logic)

Syntax	`<root entry>.<association name>[.<path expression>];`
Description	The association access is a special form of the path expression. Associations can be defined to be optional and the multiplicity can be [0,1], [1,1] or [1,n]. By following an association, you are able to read data that has been modified but not yet been stored in the database. There are the following categories of associations: Compositions, intra-business-object associations, cross-business-object associations and cross-deployment-unit associations. Compositions are automatically generated associations implemented to allow navigation of the business object node structure (that is, `ToParent`, `ToRoot`, navigation to subnodes). Due to the generated association implementation, compositions support the [1,n] multiplicity. Intra-business-object associations link nodes within one business object. Cross-business-object associations link nodes of separate business objects. Intra and cross-business-object associations can only have the [1,1] multiplicity. **i Note** To exchange data between business objects that are located in different deployment units, configure an internal communication by using the *Service Integration Wizard*. For more information, see Configure Internal Communication [page 579]. When accessing a [1,n] composition, a collection of nodes is returned. Therefore, you cannot access a member of structured data types of the foreign business object nodes. Instead, you must use the *foreach* loop statement to access the individual instances in the returned node collection (see the second example below). If the operand on the left of an assignment is an association access path expression followed by an element, data can also be written to the foreign business object node instance. If the node type matches the target definition of the association (see the third example below), you can associate a foreign business object node instance that was just created.

Example

- Intra or cross-business-object association write access ([1,1] multiplicity):

```
this.AssocToOtherBONode.Value = 42;
```

- Composition association access ([1,n] multiplicity).

```
foreach (var item in this.Items) {
  totalValue = totalValue + item.ItemValue;
}
```

- Associate node instance to current node ([1,1] multiplicity):

```
this.AssocToBONode = <node instance>;
```

The node instance, for example, is retrieved by a query.

7.2.4.7 Business Object Lifecycle (Business Logic)

Reading a business object node instances can be performed by either following an association or executing a query. Updates are simply handled by assignment statements; the path expression on the left points to an element of a business object node instance. The changes are automatically committed to the context business object once the script file completes the execution without errors.

The following methods are available for the business object lifecycle:

- Create Instance [page 246]
- Create with Reference Action [page 247]
- Query Execution [page 260]
- Retrieve Instance [page 268]
- Delete Instance [page 248]
- Check Node Existence(IsSet) [page 243]

7.2.4.8 Check Associations (Business Logic)

Syntax	`<path expression>.<association name>.IsSet();`
Description	You can use the `IsSet ()` method for [1,1] or [0,1] associations to check whether the association was set before. Otherwise any access to an association that was not set causes a runtime error.
Example	`var targetNode = this.ToTarget;`

```
if (this.ToTarget.IsSet()) {
  this.Name = "Assoc set to: " + targetNode.Name;
} else {
  this.Name = "Assoc not set.";
}
```

7.2.4.9 Check Node Existence (Business Object)

Syntax	`<node instance>.IsSet();`
Description	This method is analogous to the `IsSet` method for checking associations. However, the `IsSet` method for checking node existence checks the node references themselves. You can use this method to check if a new node instance was successfully created.

Example	`var newBO = BOName.Create();`
	`if (!newBO.IsSet()) {`
	`  raise CreationFailed.Create("E");`
	`} else { … }`

7.2.4.10 Clear Method (Business Logic)

Syntax	`<collection> | <variable>.Clear();`
Description	You can use the *Clear* method for the following scenarios: • If you have created a collection, you can use this method to remove all rows from the collection. **i Note** This method cannot be applied to read-only collections. This is relevant for code related to form data type extensions and in case of mass-enabled script files, in which the *this* keyword represents an immutable collection of nodes. • You have declared a variable by using the *elementsof* modifier and you pass initial values to subelements of the business object node. In this case, you can use the *Clear* method to remove the initial values from the subelements.

Example	Remove all rows from a collection:
	`var coll : collectionof Opportunity.Item;`
	`coll.Clear();`
	Remove initial values from subelements:
	`var struct : elementsof MyBO;`
	`struct.ID = "3865";`
	`struct.SubID = "27";`
	`struct.Clear();`

7.2.4.11 Collection Definition (Business Logic)

Syntax	`var <variable name> : collectionof <static path expression> | <data type>;`
Description	If you define a collection explicitly, the collection's row type is inferred from the static path expression, for example, a business object node, or the row type is based on an unstructured data type.

Example	`var collection : collectionof ExampleBO.Status;`
	`var collection : collectionof elementsof Opportunity.Item`
	`var collection : collectionof LANGUAGEINDEPENDENT_Text;`
	`//To use the "LANGUAGEINDEPENDENT_Text" GDT , you need to import the AP.common.GDT namespace first`

7.2.4.12 Collection Handling (Business Logic)

A collection is a standard table that is defined on the basis of business object nodes or node elements, for example, a collection of opportunities. Collections are generated when instances of business object nodes are returned by the system. You can obtain a collection of business object nodes in several ways, for example:

- You follow [1,n] composition associations. For more information, see Association Access [page 241].
- You execute a *Create with reference* action with a collection parameter. For more information, see Create with Reference Action [page 247].
- You execute a business object node query. For more information, see Query Execution [page 260].

Besides collections generated by the system, you can also define a collection explicitly.

The collection handling methods generally apply their logic to a copy of the collection and return a modified copy of the original collection. Therefore these methods declare a collection as return value. However, the *Add* method and the *Clear* method behave in a different way. They modify the collection to which they are applied. These methods do not have any return value.

> **i Note**
>
> If you use a method to add or clear a collection or a row in a collection, this only has an impact on the collection itself. To delete or create new business object nodes, use the *Delete* or *Create* operation for business objects. For more information, see Business Object Lifecycle [page 242].

The following methods are available for collections:

- Collection Definition [page 244]
- Add to Collection [page 239]
- Clear Collection [page 243]
- Sort Collection (OrderBy) [page 271]
- Filter Collection (Where) [page 249]
- Remove from Collection [page 268]
- Remove Duplicate Rows from Collection (DistinctBy) [page 267]
- Get First or Get Last Row of Collection [page 252]

- Get Empty Row of Collection (GetEmptyLine) [page 252]
- Count Rows of Collection [page 246]

7.2.4.13 Conditional Expressions (Business Logic)

Syntax	`<arithmetic expressions> == | != | < | > | <= | >= <arithmetical expression>`
Description	Conditional expressions evaluate Boolean values. The supported operators are shown in the syntax above. Use arithmetic expressions as operands, which can also be atomic expressions, such as literals, variable identifiers, and path expressions.
Example	`if (fulfillment >= 0.8 && fulfillment < 1.0) { … }`

7.2.4.14 Continue Statement and Break Statement (Business Logic)

Syntax

```
<loop statement> {
  [ <statement>* | continue; | break; |
  <complex statement> { [ continue; | break; ] ? } ]*
}
```

Description

The *continue* statement and the *break* statement can be used within the code block of loop statements, such as *foreach* and *while* statements.

The *break* statement immediately exits the enclosing loop statement. The *continue* statement quits the current loop iteration and continues with the next iteration at the top of the code block of the enclosing loop statement. In nested loop statements, however, the *break* statement only exits the inner loop.

Usually you need to embed the *continue* statement or the *break* statement into an enclosing *if* condition in order to avoid endless loops or a forced single loop execution.

Example

```
foreach (node in query.Execute()) {
  count = count + 1;
  if (count < 3) continue;
  if (count > 5) break;
  node.SomeAction();
}
```

7.2.4.15 Count Rows (Business Logic)

Syntax	`<intVar> = <collection>.Count();`
Description	If you use the *Count* method, the system returns the number of rows in a collection.

Example	`var coll : collectionof SalesOrder.Status;`
	`var counter;`
	`counter = coll.Count();`

7.2.4.16 Create Instance (Business Logic)

Syntax	Static creation of business object (root node):
	`<business object name>.Create ([<node data>]);`
	Node creation through association that is create-enabled:
	`<this | NodeInstanceVar>.<association name>.Create ([<node data>]);`

Description	You can use the *Create* method for the following:
Declarations	• Use the method to create complete business objects (that is, the root nodes) in a static manner. • You can also use the method to create subnodes of existing business objects, for example, through composition access because this access implicitly maintains the association relationship between the parent and root node. The *Create* method binds to static member path expressions that identify a business object node. It also binds to associations that are create-enabled, for example, compositions or intra-business-object associations. In compositions, the parent node instance is automatically derived from the preceding path expression and is used for the composition maintenance. The instance-based *Create* method for subnode creation is not available for variables that are derived from association access because in this case the parent node cannot be reliably determined. • Optionally, you can also provide node data to the *Create* method as a parameter. When provided, this data is automatically copied to the newly created node instance. The parameter type is the complete element structure of the node. You can infer the node element type as part of the variable declaration by identifying a certain node through a static path expression. For more information, see Declarations [page 247]. You can also assign the data to the node instance after the node has been created. However, if you create instances of SAP business objects, for some of the business objects you must provide default values in the parameters at the time the business objects are created.

Example

```
var newBO = TestBO.Create();
var itemData : elementsof TestBO.Items;
var newItem;
```

```
itemData.ItemName = "Test";
itemData.ItemValue = 42;
newItem = this.Items.Create(itemData);
```

7.2.4.17 Create With Reference Action (Business Logic)

Syntax

```
<business object name>[.<node name>].<CWR action name>(SomeNode|
NodeCollection)([<param>]*);
```

Description

The *Create with reference* (CWR) action is a special type of action that can only be executed statically and that directly returns one or more instances of the node to which the action belongs. All CWR actions are overridden to offer two signatures, one with a single node parameter and one with a collection of nodes. CWR actions return a single node or a collection of nodes depending on the collection property of the method call parameter.

> **i Note**
>
> CWR actions can only be used by SAP business objects.

Example

```
import AP.CRM.Global;
var salesOrders = SalesOrder.QueryByElements.Execute();
var allSOsCopied = SalesOrder.Copy(salesOrders);
var singleCopiedSO;
foreach (var so in salesOrders) {
  singleCopiedSO = SalesOrder.Copy(so);
  break;
}
```

7.2.4.18 Declarations (Business Logic)

Syntax

```
var <variable name> [: collectionof? elementsof? <static path
expression> | <data type>]?[= literal | <path expression>]?;
```

Description	The implementation language is a statically-typed language that allows you to use declared variables only. You always need to declare the variable before you can use it in a statement. If you declare a variable within a code block (so-called variable scoping), that is, within the braces ({...}), the scope of this variable is limited to this code block. **i Note** Variable shadowing is not supported, that is, variables cannot reuse names that are already declared in an enclosing scope. The variable type is automatically inferred upon first assignment of the variable. You can assign an initial value either by specifying a literal or by specifying a path expression, that is, elements of other already existing variables or elements of the *this* context. Assignments use copy semantics, that is, a temporary copy is created when individual elements are accessed using the *this* context. Node variables have reference semantics. Alternatively, you can explicitly specify the variable type in the variable declaration. For this, you can use basic data types as well as global data types (GDTs). You can also derive the underlying type of a business object node by using the *elementsof* modifier followed by a static path expression that identifies a business object node. This is useful if you define initial node data that can be passed as a parameter to the *Create* operation of a business object node. Do not use the *elementsof* modifier if you want to derive the type of an element from a static path expression.

Example	`var counter = 0;`
	`var amount : Amount;`
	`//To use the "Amount" GDT , you need to import the AP.Common.GDT namespace first`
	`var strVar = "test";`
	`var strVar : LANGUAGEINDEPENDENT_Text;`
	`//To use the "LANGUAGEINDEPENDENT_Text" GDT , you need to import the AP.common.GDT namespace first`
	`var boolVar = true;`
	`var name = this.Name;`
	`var nodeData : elementsof ExampleBO.Item;`
	`var nodeData : ExampleBO.Item.Element;`
	`var coll : collectionof ExampleBO.Item;`
	`var coll : collectionof elementsof ExampleBO.Item;`

7.2.4.19 Delete Instance (Business Logic)

Syntax	`<this | NodeInstanceVar>.Delete();`
Description	The *Delete* operation, which uses the inverse semantics of the *Create* operation, can only be called in the instance-based variant and must be applied to business object nodes. The reason is that the instance to be deleted must be identified by the preceding member path expression. The *Delete* operation can also be applied to collections of business object nodes.

Example

```
foreach (var BO in TestBO.QueryByElements.Execute()) {
  if (BO.Name == "delete") {
    BO.Delete();
  }
}
```

Deletion of a collection of nodes:

```
TestBO.QueryByElements.Execute().Delete()
```

7.2.4.20 Filter Collection (Business Logic)

Syntax

```
<collection A> = <collection B>.Where(<function literal>);
```

Description

You can use the *Where* method to filter a collection in order to return only the rows that meet the criteria. This method returns a new collection, for example, collection A, which contains a subset of the source collection, for example, collection B.

Example

In the following example, the collection `collB` is the result of a query that was executed previously:

```
var mySearchValues : elementsof myBO.Item;
var collA;
mySearchValues.ItemID.content = "Bill";
collA = collB.Where(n => n.ItemID != mySearchValues.ItemID ||!
(n.ItemID.content < 1000 && n.ItemID.content == "Fred"));
```

```
var collStrings : collectionof LANGUAGEINDEPENDENT_Text;
//To use the "LANGUAGEINDEPENDENT_Text" GDT , you need to import
the AP.common.GDT namespace first
var collStringsResult;
collStringsResult = collStrings.Where( n => n == "Fred" );
collStringsResult = collStrings.Where( n =>
n.StartsWith( "Fred" ) );
```

7.2.4.21 Foreach Loop Statement (Business Logic)

Syntax

```
foreach (var? <identifier> in <path expression>) {
  <statement list>
}
```

Description

The *foreach* loop statement allows you to iterate over collections of any type.

Variables that are used in a *foreach* statement are called loop variables. They have different semantics depending on the way they are declared:

- Declaration of loop variable before the *foreach* statement:
  If a loop variable is declared before the *foreach* statement, the variable type must match the row type of the collection. In this case, the loop variable is defined in the scope outside of the *foreach* statement and can therefore also be accessed after the *foreach* statement. If the loop variable refers to a collection of data types, the variable has value semantics, that is, it is a copy of the original data of the collection element in the current iteration. Therefore the original data in the collection cannot be changed.
  If a loop variable refers to a collection of business object nodes, the variable has reference semantics, that is, you can always change the data of the collection elements.
- Explicit declaration of loop variable as part of the *foreach* statement
  If you declare a loop variable within the *foreach* statement using the *var* keyword, the scope of the variable is limited to this code block, that is, to the code within the braces of the statement. Therefore the data of the last loop iteration cannot be accessed via the variable outside of the *foreach* code block.
  A loop variable that is declared explicitly within the *foreach* statement always has reference semantics, even if the variable refers to a collection of data types. This means that the original data of the collection elements can be changed. This is particularly useful for form data type extensions in the following case: a data type with a sub-element that is a collection is passed in as the `FormData` parameter and you want to change existing data in the collection elements. The loop variable itself, however, is immutable.

The statements within the *foreach* body can make use of the loop variable to access the instance of the current iteration. For information about how to obtain collections of business object nodes, see Collection Handling [page 244] .

Example

```
var node;
foreach (node in query.Execute()) {
  count = count + 1;
  if (count < 3) continue;
  if (count > 5) break;
  node.SomeAction();
}
```

Example for form data type extension:

```
foreach (var elemData in FormData.CollectionSubElement) {
  elemData.Element = "New Value";
}
```

7.2.4.22 Function Literals (Business Logic)

Syntax	Applicable in collection handling when you use the *OrderBy* and *DistinctBy* methods:
	`<function parameter name> => <function parameter name>.<method or property of inferred type>`
	Applicable in collection handling when you use the , *Where* and *Remove* methods:
	`<function parameter name> => <function parameter name>.<method or property of inferred type> <arithmetic expression>`

Description	You can only use function literals in the following collection handling methods: Where [page 249], OrderBy [page 271], DistinctBy [page 267], and Remove [page 268]. A function literal is an expression that defines an unnamed function. Function literals use the lambda operator (=>), which reads as "goes to". The left side of the lambda operator specifies the input parameter, the type of which is inferred from the last item of the path expression. The right side of the lambda operator holds the statement. In *OrderBy* methods and *DistinctBy* methods, the statement is a path expression and in *Where* methods and *Remove* methods the statement is a comparison. In all these cases, the path expression must specify a structured element. **i Note** Do not use a function parameter name that you have already used for a declared variable. For more information, see Declarations [page 247].

Example	`var myCollection : collectionof TargetBO;`
	`myCollection = this.ToNAssoc.Where(n => n.elementX == "Yes");`
	You can achieve the same result by using a *foreach* loop statement:
	`var myCollection : collectionof TargetBO;`
	`foreach (var i in this.ToNAssoc) {`
	`  if (i.elementX == "Yes") { myCollection.Add(i); }`
	`}`

7.2.4.23 Get by Index (Business Logic)

Syntax	`[Collection].GetByIndex(index)`
Description	You can use the Get By Index function to get the index value from the collection.

Example 1	`var query= TestBO.QueryByElements;`
	`var result = query.Execute();`
	`var indexval;`
	`indexval = result.GetByIndex(5);`
	`Gets the 5th index value`

Example 2	`var query = TestBO.QueryByElements;`
	`var result = query.Execute();`
	`var indexval;`
	`var counter;`
	`counter = 0;`
	`indexval = result.GetByIndex(counter);`
	`Returns empty value as the index starts with 1`

7.2.4.24 Get First or Last Row (Business Logic)

Syntax	`<collection row> = <collection>.GetFirst();`
	
	`<collection row> = <collection>.GetLast();`

Description	Use the *GetFirst* or *GetLast* method to return the first or last row of a collection.

Example	`var coll : collectionof Opportunity.Item;`
	`var first;`
	`var last;`
	`first = collection.GetFirst();`
	`last = collection.GetLast();`

7.2.4.25 Get From DB (Business Logic)

Syntax	`<business object instance>[.<node instance>].GetFromDB();`
Description	You can use the GetFromDB function to read the data of a single node instance or a collection of node instances of a business object. You can also use it to read data of dependent object nodes such as notes, attachments as well as extension nodes. The data returned is the state of the instance as read from the database before the start of the current transaction. With this, you can figure out the changes of a business object instance at the end of a transaction and build custom logic around it.

Example	`var uuid:Opportunity.UUID;`
	
	`var oppInstance = Opportunity.Retrieve(uuid);`
	`//For a single instance`
	`var dbImage = oppInstance.GetFromDB();`
	`//For a collection`
	`var collDbImage = oppInstance.Party.GetFromDB();`

7.2.4.26 If Control Statement (Business Logic)

Syntax	`if (<logical expression>) {<statement list>}`
	
	`[ else if (<logical expression >) {<statement list>} ]*`
	
	`[ else {<statement list>} ]?`

Description	The *if* statement is a control statement. Depending on the result of the logical expression, the system executes or skips the statement block. Optionally, *else if* and *else* subclauses can be used.

Example	Bonus calculation:
	
	`if (fulfillment >= 0.8 && fulfillment < 1.0) {`
	`  bonus = this.Actual.content * 0.1;`
	`}`
	`else if (fulfillment >= 1.0) {`
	`  bonus = this.Actual.content * 0.15;`
	`else { bonus = 0; }`
	`}`

7.2.4.27 Import (Business Logic)

Syntax	`import <namespace component>[.<namespace component>]*;`

Description

The *import* keyword imports business objects or reuse libraries from namespaces that are available in the studio in the *Repository View*. Imports have to be defined at the very top of a script file. You can only import complete namespaces but no individual entities. If there are naming conflicts between different entity types in the same namespace, for example, if a reuse library has the same name as a business object, you must fully qualify the name upon usage.

The namespace syntax in the scripting language is different to the namespace representation in the *Repository View*. In the scripting language, the constant `http://sap.com/xi` prefix of the URL-based namespaces is omitted and the slash mark (/) is replaced by a period (.).

Imported business objects can be statically accessed by typing the business object name as the root identifier of a path expression. The local namespace of the solution is imported implicitly, that is, local business objects can be accessed without an *import* statement.

> **i Note**
>
> If you use more than one namespace, we recommend that you define an alias for each imported namespace and qualify business objects or data types from these namespaces by using the aliases. For examples, see Name Qualification [page 257].

Code completion is available for the namespace import, as well as for the static access of imported business objects and reuse libraries. The code completion for the namespace import always offers the required amount of subsequent path elements in order to complete the namespace so that the corresponding entities can be accessed, for example, business objects or reuse libraries.

Example

```
import ABSL;
```

```
import AP.CRM.Global as CRM;
```

7.2.4.28 Literals (Business Logic)

Syntax

```
"<literal string>" | true | false |
[+|-|? DecDigit* .? DecDigit+ [ e|E [+|-]? decDigit ]?;
```

Description	The following literals can be used: • Literal strings that are included in quotation marks (“..”) • Numeric literals, that is, positive and negative integers and floating points • The Boolean values `true` and `false` You can use a literal, for example, as an expression on the right of an assignment or as a parameter of a method call in a path expression. In a literal string, you can use a backslash (\) to escape characters, such as carriage returns or quotation marks (“). You can use the following escape sequences: • \n (new line) • \r (carriage return) • \t (tab) • \v (vertical tab) • \f (form feed) • \\ (backslash) • \” (quotation marks)

Example	`var counter = 0;`
	`var strVar = "test";`
	`var escapedStringVar = "FirstLine\nSecondLine";`
	`this.ActionName(true);`

7.2.4.29 Logical Expressions (Business Logic)

Syntax	`[!]? <conditional expression> [ && | || [!]? <conditional expression> ]*`
Description	The logical operators `AND` (&&) and `OR` (||) always evaluate the overall expression to a Boolean value. The operands themselves are conditional expressions, which in turn are comprised of arithmetic expressions, finally breaking down to atomic expressions, such as literals, variable identifiers, and path expressions. You can also use the `NOT` operator (!) combined with brackets to override the precedence default value of subexpressions. **i Note** Short circuit evaluations are not supported. That is, if in a logical AND or OR expression the value of the first operand is sufficient to determine the result of the operation, the second operand is still evaluated.
Example	`if (fulfillment >= 0.8 && fulfillment < 1.0) { … }` `check = !(this.Element > 42);`

7.2.4.30 Mass Create (Business Logic)

Syntax	`Static creation of business object (root node):<business object name>[.<node name>].CreateMass (collectionof elementsof [<node data>]);`
Description	The Mass Create function allows you to create multiple instances of a business object or a business object node. It takes the collection of the node data to be created and returns the collection of node instances.

> **i Note**
>
> For the Mass Create function, the behavior of the ESF2 and BOPF-based BOs for failed instances are different:
>
> - **ESF2**: If the creation of one of the instances fails, the entire transaction is rolled back with no instance created.
> - **BOPF**:If the creation of two instances out of 10 fails, the creation of the rest of the instances is executed. For a failed instance, the API returns an empty value. The order of the instances returned by API is the same as the input collection. For example, if the creation of the 2nd and the 8th index fails, the API returns empty values for the 2nd and the 8th index.
>
> When the API returns empty values for failed instances, use the GetByIndex () function to get the input values for the indexes of empty values.
>
> For more information, see Get by Index (Business Logic) [page 251]

Example

```
Var des:elementsof InstallationPoint.Description;
var coldes:collectionof elementsof InstallationPoint.Description;
var desres;
des.Description.content="test1";
des.Description.languageCode = "DE";
coldes.Add(des);
des.Clear();
des.Description.content = "test2";
des.Description.languageCode = "EN";
coldes.Add(des);
des.Clear();
desres = this.Description.CreateMass(coldes);
```

7.2.4.31 Name Qualification (Business Logic)

Syntax	`<object type 1>::<alias for namespace>:<path element>.<object type 2>:<path element>;` Valid entries for <object type 1>: `BusinessObject`, `Library` (= reuse library or Web service), `Message` Valid entries for <object type 2>: `Node`, `Query`, `Action`, `Association`, `Element`, `Function` (= built-in function), `LibraryFunction` (= reuse service or Web service operation)
Description	Usually, you can address the imported objects directly by their names instead of using fully qualified names. However, if more than one object type (for example, a business object node, query, association, or action) is declared with the same name, you need to fully qualify the objects in order to make them unambiguous. This is necessary, for example, in the following cases: • Two namespaces are imported that contain business objects with the same name. • You have created a business object with the same name as an SAP business object. To reference the SAP business object, you need to enter the fully qualified name of this business object including its namespace alias. Without mentioning the namespace, the system will choose your business object instead of the SAP business object. • You want to access a business object that has the same name as a reuse service or a message. • A node name in one business object is the same as a query name in the same business object on the same level. If the system discovers such a naming conflict, you can either use code completion to insert the correct path expression or enter it directly by using two colons (::).

Example	Fully qualifying names because two namespaces contain business objects with the same name:
	
	`import AP.CRM.Global as CRM;`
	`var sapSalesOrder : CRM:SalesOrder;`
	`var mySalesOrder: SalesOrder;`
	
	Fully qualifying names because there are nodes, elements or associations within the same name:
	
	`import ABSL;`
	`import AP.FinancialAccounting.Global as fin;`
	
	`var costObject : fin:FinancialAccountingViewOfCostObject;`
	`var descriptionNodeElements : elementsof`
	`fin:FinancialAccountingViewOfCostObject.Node::Description;`
	`var description :`
	`fin:FinancialAccountingViewOfCostObject.Element::Description;`
	
	`costObject = fin:FinancialAccountingViewOfCostObject.Create();`
	`descriptionNodeElements.Description.content = "A Description";`
	`descriptionNodeElements.Description.languageCode =`
	`LanguageCode.ParseFromString("EN");`
	`costObject.Association::Description.Create(descriptionNodeElements);`
	
	`description = costObject.Element::Description;`

7.2.4.32 Path Expressions (Business Logic)

Syntax	`<root item>[.<data type element> | business object node feature | method name [ ( <param>* ) ] ]*`
	
	Path expression containing a library function parameter (for example, of a reuse library service) for use in a variable declaration:
	
	`<library>.<library function>.<param>.<...>`

Description	Path expressions can be used for the following: • Node elements • Associations • Method calls • Type of variables Path expressions always start with a root item. The root item can be a variable identifier, the *this* context, a static access to an imported business object or a reuse library, as well as a message declared in the business object definition. If the root item is a variable of a structured data type, the path expression addresses subelements of this data type. If the *this* context and imported business objects are used, all supported business object node features can be addressed through the path expressions. The business object node features are element and subelement access, associations, queries, actions, including Create with reference [page 247] actions, and the Create Instance [page 246] operation and the Delete Instance [page 248] operation. If the root item is a reuse library or a Web service (`library`), you must add the reuse service or Web service operation (`library function`) and at least the parameter. If the parameter is of a structured type, the path expression can also include the elements and subelements of this type. Associations are instance-based and they can therefore only be followed by the *this* context and by variables that are typed as business object nodes. In contrast, queries are static entities that have to be retrieved and executed based on imported business objects. As an action can have the multiplicity [0,0], [1,1], or [1,n], a query can either be accessed statically or instance-based, which depends on the multiplicity. For the execution of an action, you can provide parameters to the action method call in the same way as you can provide parameters for the execution of any reuse library service. Methods and complete reuse libraries can be bound to certain data types and are then automatically offered in the code completion. In addition to actions that are defined as part of the business object model and that belong to a certain business object node, there are also various methods that are available for all unstructured data types, for example, the *ToString* method.

Example	`var product = this.ProductName;`
	
	`var y : CurrencyConversion.Add.TargetCurrencyCode;`

7.2.4.33 Process Context (Business Logic)

Syntax	`<path expression to node>.GetProcessContext().<process context field>;`
Description	Within an `.absl` file related to service integration, you can use the `GetProcessContext` method for every accessible node instance. The process context itself offers change notifications for all changed reference business object instances and all already evaluated conditions of the reference business object root node instances. For the root node of the reference business object, the process context provides the Boolean field `IsActiveInProcess`. The following change notification fields are available for all accessible nodes (including nodes that are accessed through associations): • `IsCreated` • `IsUpdated` • `IsDeleted` The following evaluated conditions are available for the root node of the reference business object: • `IsStarted` • `IsChanged` • `IsCancelled` These conditions return the default condition evaluation result for the root node of the reference business object.
Example	`var isActive = this.SubNode.GetProcessContext().IsActiveInProcess;`

7.2.4.34 Query Execution (Business Logic)

Syntax

```
<query>.Execute ([<SelectionParameters>]);
<query>.Execute (<SelectionParameters>, [SortingParameters],
[<MaximumRows>], [StartRow]);
<query>.ExecuteDataOnly ([<SelectionParameters>]);
<query>.ExecuteDataOnly (<SelectionParameters>,
[SortingParameters], [<MaximumRows>], [StartRow]);
<query>.ExecuteFromDB ([<SelectionParameters>]);
<query>.ExecuteFromDB (<SelectionParameters>, [SortingParameters],
[<MaximumRows>], [StartRow]);
<query>.Count ([<SelectionParameters>]);
```

```
<query>.CreateSelectionParams ();
<selection parameter collection>.Add(<QueryParameterPath>, <Sign>,
<Option>, <LowValue>, [<HighValue>]);
<selection parameter collection>.Add(<QueryAssociationPath>,
<NodeInstance>);
```

```
<query>.CreateSortingParams ();
<selection parameter collection>.Add(<QueryElementPath>);
<selection parameter collection>.AddDescending(<QueryElementPath>);
```

Description

You can execute queries to search and retrieve data from business object nodes that is stored in the database. A query is identified by a static path starting with the name of the business object followed by the name(s) of the node(s) and the name of the query. For more information, see Queries [page 406].

Execute Function

The `Execute` function of a query returns a collection of instances of the node to which the query is bound. Queries can be executed directly with no further parameters specified. Usually, you provide selection parameters to narrow down the result set as much as possible to the desired node instances by defining appropriate filtering criteria. The instances returned by the `Execute` function overwrites any unsaved changes performed by script files before calling `Execute`.

> **i Note**
>
> A query does not take data into account that has been created or changed but has not yet been saved to the database.

Selection Parameters

Each query defines a set of selection parameters. There may be selection parameters that are not part of a node's element structure. Selection parameters are passed as a collection to the query. You can create an empty selection parameter collection by using the `CreateSelectionParams` function defined for the corresponding query. You can also add selection parameters to the collection by using the `Add` function.

`Add` requires the following parameters:

- `QueryParameterPath` identifies a query parameter, for which you define the selection parameter. `QueryParameterPath` is a path expression that starts with the query variable describing the same query that was used to create the selection parameters. It is either followed by a parameter name or, in case the parameter is structured, by an element path. The complete path must always point to an unstructured parameter.
- `Sign` defines whether the selection shall be included in the result set (`"I"`) or excluded from the result (`"E"`).
- `Option` is an operator used to compare the values defined in the selection parameters with the actual values in the business object nodes. Possible values are:

  Possible values for the Option parameter and what they mean

`EQ` = Equal	`NE` = Not equal
`GE` = Greater or equal	`LE` = Less or equal
`GT` = Greater	`LT` = Less
`CP` = Contains pattern	`NP` = Does not contain pattern
`BT` = Between	`NB` = Not between

- `LowValue` and `HighValue` define the actual filter values. `HighValue` is optional and you can use it for options that allow for ranges, that is, `"BT"` or `"NB"`. You can either define these values by specifying literals or by supplying variables or elements of node instances (for example, by using

`this.<ElementName>`). The type of the provided values must match the type of the specified query parameter structure element. Please note that he system does not perform an implicit type conversion.
The values of the options "`CP`" and "`NP`" are patterns that can include the wildcard characters "`*`" or "`+`". You can escape a wildcard character by using the "`#`" character.

There are cases where you need to query for all instances of a business object node that have an association to a certain instance of another business node. This you can do by adding a selection parameter for the source foreign key element of the association. The source foreign key elements of custom business objects are generated implicitly by the system and cannot be accessed by the script code. However, if you need to access the source foreign key elements, you can use the association name to define a selection parameter. For this purpose, the `Add` function has an additional signature with only two parameters:

- The `QueryAssociationPath` parameter identifies an association (that is, its foreign key element) for selection. `QueryAssociationPath` is a path expression that starts with the query variable describing the same query as the one that was used to create the selection parameters. The path expression is followed by an association name. Please note that the system supports only foreign key associations defined at the same node where the query is defined.
- The `NodeInstance` parameter is an instance of the business object node the association points to.

Sorting Parameters

By default, the result of a query is **not** sorted. For queries defined on custom business objects, you can specify that the result is to be sorted by specific elements. Another option is to use the sort functions offered by the collection (see Collection Handling [page 244]). However, please take into consideration that the sorting process performed by the query is executed on the database and is therefore faster than sorting a collection in memory.

You can influence the order of the query result by providing sorting parameters. You can create an empty sorting parameter collection by using the `CreateSortingParams` function defined for the corresponding query. Sorting parameters can be added to the collection by using the `Add` or the `AddDescending` function. Both functions have the same parameter:

- The `QueryElementPath` parameter identifies an element in the query result, by which the result is to be sorted. `QueryElementPath` is a static path expression that starts with the name of the business object followed by the name(s) of the node(s) and the name of a node element. In case the element is structured, the path expression is followed by an element path. The complete path must always point to an unstructured element. The element must belong to the same node the query belongs to.

When executing the query, the system applies the sorting parameters in the same order as you have added the parameters to the sorting parameters collection. The sort order (lexical or numeric) depends on the data type of the query element.

In addition to specifying sorting parameters, you can restrict the number of instances returned by the query using the `MaximumRows` parameter. If you expect a large result set, this can improve performance provided that your solution needs to process only a small number of instances. The `StartRow` parameter defines an offset in the result data that allows you to request data starting from a row different from the first one. The first row has an offset of `0`.

ExecuteDataOnly Function

The `ExecuteDataOnly` function provides for the same parameters for selection and sorting as the `Execute` function. However, `ExecuteDataOnly` function returns a collection of business object node data instead of the business object nodes themselves. The collection returned has the same type as a variable that you declare by using `collectionof elementsof <business object node>` whereas `<business object node>` is the node to which the query belongs. This means that, in case you need to modify or execute an action or follow an association based on the query result, you must explicitly retrieve the business object nodes. Despite less flexibility, please consider the `ExecuteDataOnly` function in favor of the `Execute` function because usually the `ExecuteDataOnly` function is faster. This function also does not produce any unwanted side effects by overwriting unsaved modifications.

> **i Note**
>
> The data returned by the `ExecuteDataOnly` function cannot be used to create a business object node directly although the data type returned by this function is compatible with the type expected by the `Create` function. If you try to use the data to create a node directly , this will lead to a runtime error.

ExecuteFromDB Function

The `ExecuteFromDB` function provides the same parameters for selection and sorting as the `Execute` function. However,`ExecuteFromDB` function returns the data that is saved on the Database and not from the buffer. Hence, the data returned via this function will not contain any unsaved changes done in the current transaction.

Count Function

The `Count` function works the same way as the `Execute` function or the `ExecuteDataOnly` function. However, it just returns the number of business object nodes on the database that match the selection parameters and does not return data of the selected business object nodes.

Example

Definition of business object `QueryExample`:

```
import AP.Common.GDT as apCommonGDT;
businessobject QueryExample
{
  [AlternativeKey]
  element Number : IntegerValue;
  element NextNumber : IntegerValue;
  element Period : DatePeriod;
  element Class  : ClassificationCode; // custom code list data
type with values ALPHA and BETA
  association ToOtherBusinessObject to Target;
  action Test;
}
```

Definition of business object `Target`:

```
import AP.Common.GDT as apCommonGDT;
businessobject Target
{
  [AlternativeKey]
  element ID : ID;
}
```

Script code to demonstrate various options to execute queries:

```
import ABSL;
```

```
// get the default query of QueryExample
var query = QueryExample.QueryByElements;
```

```
// 1a) Select all instances
// type of resultData is: elementsof BusinessObject::QueryExample
// which allows for direct modifications, access to associations or
actions, for example: result.GetFirst().Test();
var result = query.Execute();
```

```
// 1b) Select data of all instances
// type of resultData is: elementsof BusinessObject::QueryExample
// which just includes business object node data and therefore is
faster than 1a)
var resultData = query.ExecuteDataOnly();
```

```
// 1c) Count all instances
var rowCoumt = query.Count();
/*
Let's assume that the following instances of the QueryExample
business object exist on the data base. The result variable as well
as the resultData variable have already been filled with data of
all instances at this point in time:
```

Number	Period.StartDate	Period.EndDate	Period.Duration	Class
4	2013-01-01	2013-02-01	P1M	ALPHA
2	2013-03-01	2013-04-01	P1M	BETA
3	2013-05-01	2013-05-10	P9D	ALPHA
1	2013-05-11	2013-05-20	P9D	BETA

```
row Count is 4
*/
// 2) Select a range of Period.StartDate
var startDate1 = Date.Create(2013, 02, 01);
var startDate2 = Date.Create(2013, 05, 01);
var selectionParams = query.CreateSelectionParams();
selectionParams.Add(query.Period.StartDate, "I", "BT", startDate1,
startDate2);
resultData = query.ExecuteDataOnly(selectionParams);
/*
```

Number	Period.StartDate	Period.EndDate	Period.Duration	Class
2	2013-03-01	2013-04-01	P1M	BETA
3	2013-05-01	2013-05-10	P9D	ALPHA

```
*/
// 3) Same as in 2) but further restricted by excluding Class BETA
selectionParams.Add(query.Class, "E", "EQ",
ClassificationCode.CO_BETA);
resultData = query.ExecuteDataOnly(selectionParams);
```

```
/*
```

Number	Period.StartDate	Period.EndDate	Period.Duration	Class
3	2013-05-01	2013-05-10	P9D	ALPHA

```
*/
```

```
// 4) Same range as for 2) but sorted by Class ascending
selectionParams.Clear();
selectionParams.Add(query.Period.StartDate, "I", "BT", startDate1,
startDate2);
var sortingParams = query.CreateSortingParams();
sortingParams.Add(QueryExample.Class);
resultData = query.ExecuteDataOnly(selectionParams, sortingParams);
/*
```

Number	Period.StartDate	Period.EndDate	Period.Duration	Class
3	2013-05-01	2013-05-10	P9D	ALPHA
2	2013-03-01	2013-04-01	P1M	BETA

```
*/
// 5) Same as in 4) but request first row only (MaxRows = 0,
FirstRow = 0 (default))
resultData = query.ExecuteDataOnly(selectionParams, sortingParams,
1);
/*
```

Number	Period.StartDate	Period.EndDate	Period.Duration	Class
3	2013-05-01	2013-05-10	P9D	ALPHA

```
*/
// 6) same as in 4) but get the 10 rows starting from the second
row (MaxRows = 10, FirstRow = 1)
resultData = query.ExecuteDataOnly(selectionParams, sortingParams,
10, 1);
/*
```

Number	Period.StartDate	Period.EndDate	Period.Duration	Class
2	2013-03-01	2013-04-10	P1M	BETA

```
*/
// 7) Get all instances of QueryExample that refer to a given
instance of business object Target
```

```
selectionParams.Clear();
var targetInstance = Target.Retrieve("ID12345");
selectionParams.Add(query.ToOtherBusinessObject, targetInstance);
resultData = query.ExecuteDataOnly(selectionParams);
```

7.2.4.35 Raise Message (Business Logic)

Syntax	`raise <message name>.Create(<severity> [, <message text variable>]*);`
Description	You can use the *raise* keyword to raise messages that have been declared in the business object definition. When you use the *Create* method, an instance of a message is created that is automatically bound to the message type. The *Create* method has one mandatory parameter: the severity. You have to supply the severity as a string: • `"E"` for errors • `"W"` for warnings • `"S"` for success messages As additional optional parameters you can supply the values for placeholder variables as declared in the message text. The type of these values is `ANY`. You do not need to import messages if they are declared in the business object definition. They are then automatically available in the code completion of the current context.
Example	`raise MissingTest.Create("E", this.ObjectID);`

7.2.4.36 Remove Duplicate Rows (Business Logic)

Syntax	`<collection A> = <collection B>.DistinctBy(<function literal>).ThenBy(<function literal>);`
Description	You can use the *DistinctBy* method to delete duplicate rows in a collection according to the specified criteria.

Example	`var collMyCampaigns : collectionof Campaign;`
	`var result;`
	`result =`
	`collMyCampaigns.DistinctBy( n=>n.Status.LifeCycleStatusCode ).ThenBy( m=>m.PlannedStartDate );`
	
	`var collStrings : collectionof LANGUAGEINDEPENDENT_Text;`
	`//To use the "LANGUAGEINDEPENDENT_Text" GDT , you need to import the AP.common.GDT namespace first`
	`var collStringsUnique;`
	`collStringsUnique = collStrings.DistinctBy( n=>n );`

7.2.4.37 Remove from Collection (Business Logic)

Syntax	`<collection A> = <collection B>.Remove(<function literal>);`
Description	You can use the *Remove* method to remove those rows from the collection that meet the criteria. This method returns a new collection, for example, collection A, which contains a subset of the source collection, for example, collection B.

Example	In the following example, the collection `collB` is the result of a query that was executed previously :
	`var mySearchValues : elementsof myBO.Item;`
	`var collA;`
	`mySearchValues.ItemID.content = "Smith";`
	`collA = collB.Remove(n => n.ItemID != mySearchValues.ItemID &&`
	`n.ItemID.content != "Miller");`

7.2.4.38 Reset Associations (Business Object)

Syntax	`<path expression>.<association name>.Reset();`
Description	This method allows you to reset an association. If, for example, a target node instance has been deleted by following the association, you must reset the association because the references that point to the deleted business object are not cleared automatically. Therefore, such invalid references must be explicitly deleted and every access to an association must be secured by a previous `IsSet` check call. For information about the `IsSet` check for associations, see Check Associations [page 242]. The `Reset` method can only be applied to foreign key associations and is not available for variables that are derived from association access.

Example	`var targetNode = this.ToTarget;`
	`if (targetNode.IsSet()) {`
	`  targetNode.Delete();`
	`  this.ToTarget.Reset();`
	`}`

7.2.4.39 Retrieve Instance (Business Logic)

Syntax	`<node instance> = <business object>.Retrieve(<alternative key>);` `<collection of node instances> = <business object>.Retrieve(<collection of alternative keys>);`

Description	You can use the *Retrieve* method to read the data of a single node instance or a collection of node instances of a business object. The *Retrieve* method enables you to read data that has been modified but not yet been stored in the database. As parameters for the method, you can pass a single alternative key or a collection of alternative keys, which have been declared as variables. The variables, which contain the alternative key values, need to be of the same type as the alternative keys specified in the business object. If the method is used for a collection of node instances, it returns a new collection.

Example	Definition of alternative key in business object:
	
	`businessobject BonusPlan {`
	`  [AlternativeKey] element SO_ID : BusinessTransactionDocumentID;`
	`}`
	
	Script file:
	
	`var elementID : xyz.SO_ID;`
	`var collectionID : collectionof xyz.SO_ID;`
	`var retrieveColl;`
	`var retrieveSingle;`
	
	`elementID.content = "5000000002";`
	
	`retrieveSingle = SalesOrder.Retrieve(elementID);`
	
	`collectionID.Add(elementID);`
	`elementID.content = "5000000003";`
	`collectionID.Add(elementID);`
	
	`retrieveColl = SalesOrder.Retrieve(collectionID);`

7.2.4.40 Return Values (Business Logic)

Syntax	`return <literal> | <path expression>;`
Description	The use of the *return* keyword is mandatory in `.absl` files that were created by using the *Service Integration* wizard. The return value has to be of Boolean type. For each of the four condition evaluations (Relevance, Start, Change, Cancel) that are possible in a service integration scenario, the Boolean return value must be used to signal the calling process agent about the evaluation result. To calculate the condition for the current root node instance, you can access the process context information of this node and its node values.
Example	`return true;` `return this.Count > 7;`

7.2.4.41 Script File Parameter (Business Logic)

Description	The *InReconciliation* Boolean indicator specifies conditions that depend on the mode of execution, that is, the standard message mode or the reconciliation message mode. The reconciliation mode is an error recovery mechanism that allows it to send a new message, the reconciliation message. This reconciliation message contains the complete information and cancels all previous erroneous messages.

Example	`if (InReconciliation) {`
	`...`
	`}`
	`return false;`

7.2.4.42 Service Integration Methods (Business Logic)

Service integration allows you to synchronize data between several involved business objects that are located in different deployment units in an asynchronous way. The business objects exchange the data through XML messages. You can use the methods listed below to determine for which nodes of the sending business object the data is transmitted.

Script files can be used for the condition evaluation, which specifies whether the data of a node is part of the cross-business-object synchronization. Therefore, the `.absl` files in service integration scenarios support parameters that are passed in at runtime. Additionally, a Boolean return value is expected that signals to the calling process agent whether the business object instance is to be considered in the message construction. Furthermore, the process context is attached to every business object node instance that allows you to check certain status information related to service integration.

In script files that are executed within a service integration scenario, the context information is read-only. This means that the system prevents accessible business object nodes and calls of actions from being changed. The information about the script file signature (that is, the parameters and return values and their corresponding types) is documented in a comment block at the very beginning of each `.absl` file. In service integration scenarios, various `.absl` file subtypes are involved. This is also documented in the comment.

The following methods are available for service integration:

- Script File Parameter [page 270]
- Return Values [page 269]
- Process Context [page 259]

7.2.4.43 Set Associations (Business Logic)

Syntax	`<path expression>.<association name> = <path expression>;`

Description	You can navigate along associations that have been set by the user interface but you can also set associated node instances explicitly. For this, you can use a simple standard assignment expression that on the left provides a path expression, whose last element is an association. The path expression on the right must evaluate to the node type that is expected to be the target node of the modeled association. For such assignments, you can only use modeled associations with [1,1] or [0,1] multiplicity. The expression on the right can be, for example, an item of a query execution result or a *Create* operation.

Example	• Assign node instance retrieved by query execution:
	`foreach (var item in query.execute(selParams)) { this.AssocToTarget = item; break; }`
	• Assign newly created node instance:
	`var data : elementsof TargetBO; data.Name = "Created"; this.ToTarget = TargetBO.Create(data);`

7.2.4.44 Sort Collection (Business Logic)

Syntax	`<collection A> = <collection B>.OrderBy(<function literal>).ThenByDescending(<function literal>).ThenBy ...;` `<collection A> = <collection B>.OrderByDescending(<function literal>).ThenBy(<function literal>).ThenBy ...;`
Description	You can use the *OrderBy* or *OrderByDescending* method to sort a collection in ascending or descending order according to the specified sorting criteria. By default, collection columns of numeric and byte type are sorted by their value and columns of string type are sorted as text. You can define the sorting criteria as action parameters by using a function literal, which refers to the columns of the collection. To define multiple sorting criteria, add *ThenBy* or *ThenByDescending* statements to the expression. **i Note** If the row type of a collection is structured, you need to specify the sorting criteria. The sorting parameters must be an atomic data type, that is, you are not allowed to define a structured attribute of the collection row type as a sorting parameter. If the row type is unstructured, you can only sort by row and cannot specify a sorting criteria. Sorting does not have a permanent effect on business object nodes. Each time a node is read from the system, all previous sorting results are lost.

Example

```
var collItems : collectionof Opportunity.Item;
var collItemsSorted : collectionof Opportunity.Item;
collItemsSorted =
collItems.OrderBy(n=>n.NetAmount.currencyCode).ThenBy(n=>m.NetAmount
.content);
```

```
var collStrings : collectionof String;
var collStringSorted;
collStringSorted = collStrings.OrderByDescending( n => n );
```

7.2.4.45 Switch Control Statement (Business Logic)

Syntax

```
switch ( <expression> ) {
  case <expression>) [, <expression>* ] { <statement list> }
  [ default { <statement list>} ] ?
}
```

Description

The *switch* control statement provides a more convenient option to express `if.. else if..` code blocks that evaluate exactly the same field against different values again and again. The *switch* type is limited to non-structured data types. The *case* clause allows for arbitrary expressions, for example, the use of variables.

The *default* clause is optional.

Example

```
switch (this.Amount.content) {
  case  1, 2 {this.Text = "alt1"; }
  case varName.element {this.Text = "alt2"; }
  default { this.Text = "initial"; }
}
```

7.2.4.46 This (Business Logic)

Syntax

```
this.<feature name>;
```

Description

The *this* keyword defines the context in which the script file is executed. You must use the keyword as the first member of a path expression that accesses the context. The keyword has the type of the business object node that the script file is assigned to and, at runtime, points to the current instance of such a business object node. The keyword can define the root of a path expression, just as imported business objects do for static access. The *this* keyword itself is immutable and cannot be modified, whereas the elements of the node represented by *this* can be modified.

> **i Note**
>
> If you use the *this* keyword in the coding of a mass-enabled script file, *this* represents a collection of business object nodes.

The code completion for the business object node specified by *this* displays instance-based features such as node elements, associations and instance-based actions including mass-enabled features. Queries and static actions are not instance-based and therefore you can address them only through static business object access.

You can use the `GetObjectNodeReference()` method along with the *this* keyword to return the following information about the business object nodes in a solution:

- Node ID of the relevant business object node
- ObjectTypeCode of the business object
- ObjectNodeTypeCode of the node

Example

```
var name = this.Name;
```

```
var ObjNodeRef = this.Item.GetObjectNodeReference();
```

7.2.4.47 While Loop Statement (Business Logic)

Syntax

```
while (<logical expression>) {
  <statement list>
```

```
}
```

Description

The *while* loop statement allows you to execute the statement list in the code body as long as the logical expression evaluates to true.

Example

```
while (this.StrElement.Length() < 10) {
  this.StrElement = this.StrElement + "Test";
  if (this.StrElemen.Length() > 7) break;
}
```

7.2.5 SAP Reuse Libraries

7.2.5.1 Basic SAP Reuse Libraries

7.2.5.1.1 Basic SAP Reuse Libraries Reference

The basic SAP reuse libraries of the scripting language extend the scope and functions that you use to implement your business logic in the studio. These “language-near” libraries provide basic functions that are used quite often, for example, to retrieve context data, such as the current date or time for the current identity.

SAP reuse libraries allow for parameters that are either data types (CDTs or GDTs) or business object node instances. The return values have to be data types.

> **i** Note
>
> To use basic SAP reuse libraries, you need to import the namespace *ABSL* with the *import* keyword. For more information on the *import* keyword, see Syntax for Implementation of Business Logic [page 235].

The functions of the SAP reuse libraries can always be accessed in a static manner. In addition, most of the libraries are bound to a certain data type. In this case, the instance of the data type on which the method is executed will be automatically passed to the method at runtime as the first parameter.

> **Example**
>
> ```
> import ABSL;
> var globalDateTime = GlobalDateTime.ParseFromString("2008-03-30");
> var date = globalDateTime.ConvertToDate():
> ```

List of Basic SAP Reuse Libraries

- Context [page 276]
  Provides context information, for example, the current time in the time zone of the user or the identity UUID of the user that is currently logged on.
- Duration [page 287]
  Compares durations and performs simple calculations with durations.
- Numeric [page 298]
  Provides basic functions for integers and decimals.
- Numeric Character [page 299]
  Converts a string into a numeric character.
- Currency Conversion [page 279]
  Provides functions for currency-based amounts. For example, you can convert an amount from one currency into another.
- Quantity Conversion [page 300]
  Compares quantities and performs simple calculations with quantities.
- UUID [page 304]
  Provides basic functions for UUIDs.

- Language Code [page 296]
  Converts a string into a language code.
- Date [page 283]
  Compares dates, performs simple calculations with dates, and provides further useful date functions.
- Time [page 302]
  Compares times and provides further useful time functions.
- Date and Time [page 287]
  Converts a string into a date and time value.
- Global Date and Time [page 290]
  Compares global date and time values, performs simple calculations with global date and time values, and provides further useful global date and time functions.
- Local Date and Time [page 296]
  Converts local date and time values.
- Local Normalised Date and Time [page 297]
  Converts local normalized date and time values.
- Web Service Utilities [page 307]
  Calls REST mashup Web services.
- Binary [page 275]
  Converts a string to a binary value.
- Number Range [page 297]
  Draws numbers from a number range for different object types.
- Change History [page 282]
  Retrieves data from a supplied Business Object and its nodes that was changed during the given period based on the Platform Change Document service.
- Sensitive Personal Data [page 304]
  Writes direct access logs when sensitive personal data is read and sent through one of the channels.

7.2.5.1.2 Binary (Reuse Library)

This SAP reuse library contains a function that you can use to convert a string to a binary value.

Binary.ParseFromString

Description	Converts a string to a binary value using UTF-8 encoding. **i Note** This function supports static calls only.
Example	`this.Binary.content = Binary.ParseFromString("This is a test string.");` **Result:** Binary value`54686973206973206120746573742073747269E672E`

Binary.ParseFromBase64String

Description	Converts a Base64 string to a binary value using UTF-8 encoding. **i Note** This function supports static calls only.
Example	`this.Binary.Content = Binary.ParseFromBase64String("This is a Base64 string");` **Result:** Binary value `4E18AC8AC6816AC7BAE2CB6B8A78`

Binary.ToBase64String

Description	Converts a binary value to a Base64 string using UTF-8 encoding. **i Note** This function supports static calls only.

Example	`var bin = Binary.ParseFromString("This is a string");`
	
	Result: Binary value `5468697320697320612073747269E67`
	
	`var base = Binary.ToBase64String(bin);`
	
	Result: Base64 string `VGhpcyBpcyBhIHN0cmluZw==`

7.2.5.1.3 Context (Reuse Library)

This SAP reuse library contains functions that provide context information, for example, the current time in the time zone of the user or the identity UUID of the user that is currently logged on.

i Note

The *Context* reuse library supports static calls only.

Context.GetCurrentGlobalDateTime

Description	Returns the current date and time in Coordinated Universal Time (UTC).
Example	`this.result = Context.GetCurrentGlobalDateTime();` **Result:** Current date and time in UTC

Context.GetCurrentIdentityUUID

Description	Returns the Identity UUID of the user currently logged on.

Example	`this.result = Context.GetCurrentIdentityUUID();`
	Result: Identity UUID of the user currently logged on.

Context.GetCurrentSystemDate

Description	Returns the current date in the time zone of the system.
Example	`this.result = Context.GetCurrentSystemDate();`
	Result: Current date in the time zone of the system

Context.GetCurrentSystemDateTime

Description	Returns the current date and time in the time zone of the system.
Example	`this.result = Context.GetCurrentSystemDateTime();`
	Result: Current date and time in the time zone of the system.

Context.GetCurrentSystemTime

Description	Returns the current time in the time zone of the system.
Example	`this.result = Context.GetCurrentSystemTime();`
	Result: Current time in the time zone of the system.

Context.GetCurrentUserDate

Description	Returns the current date in the time zone of the user.
Example	`this.result = Context.GetCurrentUserDate();`
	Result: Current date in the time zone of the user

Context.GetCurrentUserDateTime

Description	Returns the current date and time in the time zone of the user.
Example	`this.result = Context.GetCurrentUserDateTime();`
	Result: Current date and time in the time zone of the user

Context.GetCurrentUserTime

Description	Returns the current time in the time zone of the user.
Example	`this.result = Context.GetCurrentUserTime();`
	Result: Current time in the time zone of the user

Context.GetCurrentUserLanguage

Description	Returns the language specified for the user.
Example	`this.result = Context.GetCurrentLanguage();` **Result:** Language specified for the user.

Context.GetCurrentUISwitches

Description	Returns the user interface switches (UI switches) of the current solution assigned to the current user.
Example	`this.result = Context.GetCurrentUserUISwitches();` **Result:** Table listing the UI switches. The line type of the table is PDI_ABSL_STRING

Context.GetCurrentSystemTimeStamp

Description	Returns the time stamp for the current Coordinated Universal Time (UTC) reference time of the system as a numeric value.
Example	`this.result = Context.GetCurrentSystemTimeStamp();` **Result:** Time stamp (numeric value) for the current UTC reference time of the system, for example, `20130314101211.5922640`.

Context.GetSystemURL

Description	Returns the URL of the system. If the optional parameter *SingleSignOn* is set to *True*, the SSO enabled URL is returned.

Example	`this.normal_url = Context.GetSystemURL();`
	`this.sso_url = Context.GetSystemURL(true);`
	
	Result: Normal URL of the system, for example, `https://my012345.sapbydesign.com`.
	
	OR
	
	Single Sign On Enabled URL of the system, for example, `https://my012345-sso.sapbydesign.com`.

Context.IsProductionTenant

Description	Returns True or False depending on the tenant as follows: • True for Production tenant • False for others
Example	`this.result = Context.IsProductionTenant();` **Result:** True or False depending on the tenant.

Context.GetAssignedBusinessRoles

Description	Returns all the assigned business roles for a logged in user.

Example	`var assignedRolesCollection: Collectionof IdentityBusinessRoleID;`
	`var assignedRoles:IdentityBusinessRoles;`
	`assignedRoles = Context.GetAssignedBusinessRoles();`
	`assignedRolesCollection = assignedRoles.BusinessRoles;`
	
	Result: Assigned business roles, for example: Sales Manager, Administrator.

Context.IsPatchSolution

Description	Returns True or False depending on the solution as follows: • True for Patch solution • False for others
Example	`this.result = Context.IsPatchSolution();` **Result:** True or False depending on the solution.

7.2.5.1.4 Currency Conversion (Reuse Library)

This SAP reuse library contains functions that you can use for currency-based amounts. For example, you can convert an amount from one currency into another.

CurrencyConversion.GreaterEquals

Description	Indicates whether or not an amount is greater than or equal to another amount. You can compare amounts with different currency codes. The result depends on the exchange rate maintained in the system.

Example	`this.amount1.content      = 10;`
	`this.amount1.currencyCode = "EUR";`
	`this.amount2.content      = 23;`
	`this.amount2.currencyCode = "EUR";`
	`this.result = this.amount1.GreaterEquals(this.amount2);`
	
	Result: `False`

CurrencyConversion.GreaterThan

Description	Indicates whether or not an amount is greater than another amount. You can compare amounts with different currency codes. The result depends on the exchange rate maintained in the system.

Example

```
this.amount1.content      = 10;
this.amount1.currencyCode = "EUR";
this.amount2.content      = 23;
this.amount2.currencyCode = "EUR";
this.result = this.amount1.GreaterThan(this.amount2);
```

Result: `False`

CurrencyConversion.LessEquals

Description

Indicates whether or not an amount is less than or is equal to another amount.

You can compare amounts with different currency codes. The result depends on the exchange rate maintained in the system.

Example

```
this.amount1.content      = 10;
this.amount1.currencyCode = "EUR";
this.amount2.content      = 23;
this.amount2.currencyCode = "EUR";
this.result = this.amount1.LessEquals(this.amount2);
```

Result: `True`

CurrencyConversion.LessThan

Description

Indicates whether or not an amount is less than another amount.

You can compare amounts with different currency codes. The result depends on the exchange rate maintained in the system.

Example

```
this.amount1.content      = 10;
this.amount1.currencyCode = "EUR";
this.amount2.content      = 23;
this.amount2.currencyCode = "EUR";
this.result = this.amount1.LessThan(this.amount2);
```

Result: `True`

CurrencyConversion.Equals

Description

Indicates whether or not an amount is equal to another amount.

You can compare amounts with different currency codes. The result depends on the exchange rate maintained in the system.

Example

```
this.amount1.content      = 10;
this.amount1.currencyCode = "EUR";
this.amount2.content      = 23;
this.amount2.currencyCode = "EUR";
this.result = this.amount1.Equals(this.amount2);
```

Result: `False`

CurrencyConversion.Add

Description	Adds an amount to another amount. You can add amounts with different currency codes and define a target currency code. The result depends on the exchange rate maintained in the system.

Example	`this.amount1.content = 10;`
	`this.amount1.currencyCode = "EUR";`
	`this.amount2.content = 23;`
	`this.amount2.currencyCode = "EUR";`
	`this.result = this.amount1.Add(this.amount2);`
	
	Result: Amount value `33 EUR`

CurrencyConversion.Subtract

Description	Subtracts an amount from another amount. You can add amounts with different currency codes and define a target currency code. The result depends on the exchange rate maintained in the system. Negative results are allowed.

Example	`this.amount1.content = 10;`
	`this.amount1.currencyCode = "EUR";`
	`this.amount2.content = 23;`
	`this.amount2.currencyCode = "EUR";`
	`this.result = this.amount1.Subtract(this.amount2);`
	
	Result: Amount value `-13 EUR`

CurrencyConversion.Round

Description	Rounds an amount to two decimals.

Example	`this.amount.content = 10.3754;`
	`this.amount.currencyCode = "EUR";`
	`this.result = this.amount.Round();`
	
	Result: Amount value `10.38 EUR`

CurrencyConversion.ConvertCurrency

Description	Converts an amount into a different currency based on the exchange rate maintained in the system.

Example	`this.amount.content = 10.3754;`
	`this.amount.currencyCode = "EUR";`
	`this.result = this.amount.ConvertCurrency("USD");`
	
	Result: Depends on the exchange rate

CurrencyConversion.SpellAmount

Description	Spells out an amount. The integer part and the decimal part of a decimal number are spelled out separately and returned in the respective fields. If you do not provide the language, the logon language is used as a default.

Example

```
var spell;
this.Amount1.content = 10.14;
this.Amount1.currencyCode = "EUR";
this.Language = "EN";
spell = this.Amount1.SpellAmount (this.Language);
this.Integer = spell.IntegerValue;
this.Decimals = spell.DecimalPart;
```

Result: Integer value: “TEN”, decimal part: “FOURTEEN”

7.2.5.1.5 Change History (Reuse Library)

This SAP reuse library contains a function that you can use to retrieve data from a supplied Business Object and its nodes that was changed during the given period based on the Platform Change Document service.

> **i** Note
>
> - Only the data of those elements and nodes of Custom Business Objects can be retrieved which are annotated with Change History (Business Object) [page 199].
> - The data of SAP Business Objects will also be retrieved if those business objects have the Platform Change Document service enabled by SAP.
> - This annotation can also be applied to extension nodes and their elements. However, it cannot be applied to Extension Elements.
> - This annotation does not work for Dependent Objects.

Change History. Read

Description

Returns the data from the Platform Change Documents that matches the selection criteria as described in the note above.

- *BusinessObjectName* defines the name of the Business Object. If this is a SAP Business Object it must contain the respective. namespace, e.g,. **AP.CRM.Global:Lead**.
- *NodeID* provides the NodeID of the Root node for the requested instance.
- The other four parameters are optional:
  - *NodeName*: Restricts the result to given node
  - *ToChangeDateTime*: If no value is given the current date and time is taken
  - *FromChangeDateTime*: If no value is given 30 days earlier than the *ToChangeDateTime* is used
  - *ChangerUUID*: Only the changes made to the respective. user are returned

Example for SAP Business Object	`// get all changes of SAP Business Object Lead based upon Lead Root node ID`
	`// for node Root and date from elements of current node`
	`// made by own user`
	`var  myLeadRootNodeId :  UUID;`
	`myLeadRootNodeId = …`
	`var   currentUser  = Context.GetCurrentIdentityUUID();`
	`var myChanges = ChangeHistory.Read( "AP.CRM.Global:Lead",`
	`myLeadRootNodeId,`
	`"Root",`
	`this.FromDateTime,`
	`this.ToDateTime,`
	`currentUser );`

Example for Custom Business Object	`// get all changes of Custom Business Object MyOwnBusinessObject based upon own BO Root node ID`
	`// for node Root and the last month`
	`// made by anybody`
	`var  currentUser = Context.GetCurrentIdentityUUID();`
	`var  myChanges = ChangeHistory.Read("MyOwnBusinessObject",`
	`this.GetObjectNodeReference().UUID`
	`"Root");`

7.2.5.1.6 Date (Reuse Library)

This SAP reuse library contains functions that you can use to compare dates and to perform simple calculations with dates. Furthermore, it contains other useful date features, for example, services that you can use to determine the start or end date of a month related to a specified date.

Date.Create

Description	Creates a date. **i Note** This function supports static calls only.
Example	`this.result = Date.Create(30,3,2008);` **Result:** Date value `20080330`

Date.GreaterEquals

Description	Indicates whether or not a date is later than or equal to another date.

Example	`this.date1 = Date.ParseFromString("20100215");`
	`this.date2 = Date.ParseFromString("20100402");`
	`this.result = this.date1.GreaterEquals(this.date2);`
	Result: `False`

Date.GreaterThan

Description	Indicates whether or not a date is later than another date.

Example	`this.date1 = Date.ParseFromString("20100215");`
	`this.date2 = Date.ParseFromString("20100402");`
	`this.result = this.date1.GreaterThan(this.date2);`
	
	Result: `False`

Date.LessEquals

Description	Indicates whether or not a date is earlier than or equal to another date.

Example	`this.date1 = Date.ParseFromString("20100215");`
	`this.date2 = Date.ParseFromString("20100402");`
	`this.result = this.date1.LessEquals(this.date2);`
	
	Result: `True`

Date.LessThan

Description	Indicates whether or not a date is earlier than another date.

Example	`this.date1 = Date.ParseFromString("20100215");`
	`this.date2 = Date.ParseFromString("20100402");`
	`this.result = this.date1.LessThan(this.date2);`
	
	Result: `True`

Date.IsBetween

Description	Indicates whether or not a date is between two specified date values.

Example	`this.date1 = Date.ParseFromString("20100215");`
	`this.date2 = Date.ParseFromString("20100330");`
	`this.date3 = Date.ParseFromString("20100402");`
	`this.result = this.date2.IsBetween(this.date1, this.date3);`
	
	Result: `True`

Date.GetDay

Description	Returns the day of a date.

Example	`this.date = Date.ParseFromString("20100215");`
	`this.result = this.date.GetDay();`
	
	Result: String value `15`

Date.GetWeekday

Description	Returns the day of the week of a date.

Example	`this.date = Date.ParseFromString("20100215");` `this.result = this.date.GetWeekday();`
	Result: Integer value `1` to `7`

Date.GetMonth

Description	Returns the month of a date.
Example	`this.date = Date.ParseFromString("20100215");` `this.result = this.date.GetMonth();`
	Result: String value `02`

Date.GetYear

Description	Returns the year of a date.
Example	`this.date = Date.ParseFromString("20100215");` `this.result = this.date.GetYear();`
	Result: String value `2010`

Date.GetWeek

Description	Returns the calendar week of a date.
Example	`this.date = Date.ParseFromString("20100215");` `this.result = this.date.GetWeek();`
	Result: String value `07`

Date.AddDuration

Description	Adds a duration to a date.

Example	`this.date = Date.ParseFromString("20100215");`
	`this.result =`
	`this.date.AddDuration(Duration.ParseFromString("P3D")););`

	Result: Date value `20100218`

Date.SubtractDuration

Description	Subtracts a duration from a date.

Example	`this.date = Date.ParseFromString("20100215");`
	`this.result =`
	`this.date.SubtractDuration(Duration.ParseFromString("P3D")););`

	Result: Date value `20100212`

Date.Delta

Description	Returns the delta duration between two specified date values.

Example	`this.StartDate = Date.ParseFromString("20150214");`
	`var OneMonthLater =`
	`this.StartDate.AddDuration(Duration.ParseFromString(“P1M”);`
	`this.DeltaAsMonth = this.StartDate.Delta(OneMonthLater);`
	`this.DeltaAsDays = this.StartDate.Delta(OneMonthLater,true);`
	
	Result:
	
	Duration value of DeltaAsMonth =`P1M`, calculated as 30 days
	
	Duration value of DeltaAsDays =`P28D`, calculated as 28 days

Date.StartOfMonth

Description	Returns the start of the month related to a specified date.

Example	`this.date = Date.ParseFromString("20100215");`
	`this.result = this.date.StartOfMonth();`
	
	Result: Date value `20100201`

Date.EndOfMonth

Description	Returns the end of the month related to a specified date.

Example	`this.date = Date.ParseFromString("20100215");`
	`this.result = this.date.EndOfMonth();`
	
	Result: Date value `20100228`

Date.StartOfYear

Description	Returns the start of the year related to a specified date.

Example	`this.date = Date.ParseFromString("20100215");`
	`this.result = this.date.StartOfYear();`
	
	Result: Date with value `20100101`

Date.EndOfYear

Description	Returns the end of the year related to a specified date.

Example	`this.date = Date.ParseFromString("20100215");`
	`this.result = this.date.EndOfYear();`
	
	Result: Date with value `20101231`

Date.ConvertToGlobalDateTime

Description	Converts a date into a global date and time value.

Example	`this.date = Date.ParseFromString("20100215");`
	`this.result = this.date.ConvertToGlobalDateTime();`
	
	Result: Global date time value `20100215000000`

Date.ParseFromString

Description	Converts a string into a date. The service accepts both the internal format and the ISO format. **i Note** This service only supports static calls.
Example	`this.date = Date.ParseFromString("20100215");` **Result:** Date value `20100215`

7.2.5.1.7 Date and Time (Reuse Library)

This SAP reuse library contains a function that you can use to convert a string into a date and time value.

DateTime.ParseFromString

Description	Converts a string into a date and time value. **i Note** This function only supports static calls. Use this function only when an SAP business object uses the data type `DateTime` in namespace `http://sap.com/xi/BASIS/Global`. However, do not use the `DateTime` data type in your own business object definitions.
Example	`this.dateTime = DateTime.ParseFromString("2008-03-30T10:31:00Z");` **Result:** Date and time value `2008-03-30T10:31:00Z`

7.2.5.1.8 Duration (Reuse Library)

This SAP reuse library contains functions that you can use to compare durations and to perform simple calculations with durations.

i Note

Durations are represented as `W3C xsd:duration` according to ISO 8601.

Duration.Create

Description	Creates a duration. **i Note** This function supports static calls only.
Example	`this.result = Duration.Create(1,0,14,0,0,0);` **Result:** Duration value `P1Y14D`

Duration.GreaterEquals

Description	Indicates whether or not a duration is greater than or equal to another duration.

Example	`this.duration1 = Duration.ParseFromString("P3Y");`
	`this.duration2 = Duration.ParseFromString("P5Y");`
	`this.result = this.duration1.GreaterEquals(this.duration2);`
	
	Result: `False`

Duration.GreaterThan

Description	Indicates whether or not a duration is greater than another duration.

Example	`this.duration1 = Duration.ParseFromString("P3Y");`
	`this.duration2 = Duration.ParseFromString("P5Y");`
	`this.result = this.duration1.GreaterThan(this.duration2);`
	
	Result: `False`

Duration.LessEquals

Description	Indicates whether or not a duration is less than or equal to another duration.

Example	`this.duration1 = Duration.ParseFromString("P3Y");`
	`this.duration2 = Duration.ParseFromString("P5Y");`
	`this.result = this.duration1.LessEquals(this.duration2);`
	
	Result: `True`

Duration.LessThan

Description	Indicates whether or not a duration is less than another duration.

Example	`this.duration1 = Duration.ParseFromString("P3Y");`
	`this.duration2 = Duration.ParseFromString("P5Y");`
	`this.result = this.duration1.LessThan(this.duration2);`
	
	Result: `True`

Duration.AddDuration

Description	Adds a duration to another duration. The result is not normalized. Normalization can be forced by setting the parameter `ForceNormalization` to `True`.

Example

```
this.duration1 = Duration.ParseFromString("P28DT1H");
this.duration2 = Duration.ParseFromString("P10DT2H");
this.result = this.duration1.AddDuration(this.duration2);
```

Result: Duration value `"P38DT3H"`

```
this.result = this.duration1.AddDuration(this.duration2, true);
```

Result: Duration value `P1M8D`

Duration.SubtractDuration

Description	Subtracts a duration from another duration. The subtraction may not have an unambiguous result (the reason is, for example, that some months have 30 days, others have 31 days). In this case, the function will not return a result . The calculation can be forced by setting parameter `ForceCalculation` to `True`. Then the function, for example, constantly uses 30 days for the duration of each month.

> **i Note**
>
> Negative durations are not allowed. Therefore, the first duration value must be greater than the second duration value.

Example

```
this.duration1 = Duration.ParseFromString("P1M8D");
this.duration2 = Duration.ParseFromString("P10D");
this.result = this.duration1.SubtractDuration(this.duration2);
```

Result: As the result is ambiguous the function does not return a result. Instead it returns a corresponding error message.

```
this.result = this.duration1.SubtractDuration(this.duration2, true);
```

Result: The function uses 30 days for the duration of each month. The result is the duration value `P28D`

Duration.ConvertToDays

Description	Converts a duration into days.

Example

```
this.duration = Duration.ParseFromString("P1Y14D");
this.result = this.duration.ConvertToDays();
```

Result: Integer value `379`

Duration.ConvertToHours

Description	Converts a duration into hours.

Example	`this.duration = Duration.ParseFromString("P1Y14D");` `this.result = this.duration.ConvertToHours();`
	Result: Integer value `9101`

Duration.ParseFromString

Description	Converts a string into a duration
	i Note This function supports static calls only.
Example	`this.duration = Duration.ParseFromString("P1Y14D");`
	Result: Duration value `P1Y14D`

7.2.5.1.9 Global Date and Time (Reuse Library)

This SAP reuse library contains functions that you can use to compare global date and time values and to perform simple calculations with global date and time values. Furthermore, it contains other useful global date and time features, for example, functions that you can use to determine the start or end of a global date and time of a month.

i Note

For *GlobalDateTime* elements, use the data type `GLOBAL_DateTime` of namespace `http://sap.com/xi/BASIS/Global`.

Some functions include the `IsUTC` parameter. This parameter indicates whether global date and time is passed in Coordinated Universal Time (UTC) or to the time zone of the user. This parameter is needed because the user interface does not pass the date and time in UTC but in the time zone of the user.

GlobalDateTime.GreaterEquals

Description	Indicates whether or not a global date and time value is greater than or equal to another global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.gDateTime2 = GlobalDateTime.ParseFromString("20100402235401");`
	`this.result = this.gDateTime1.GreaterEquals(this.gDateTime2);`

	Result: `False`

GlobalDateTime.GreaterThan

Description	Indicates whether or not a global date and time value is greater than another global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.gDateTime2 = GlobalDateTime.ParseFromString("20100402235401");`
	`this.result = this.gDateTime1.GreaterThan(this.gDateTime2);`
	
	Result: `False`

GlobalDateTime.LessEquals

Description	Indicates whether or not a global date and time value is less than or equal to another global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.gDateTime2 = GlobalDateTime.ParseFromString("20100402235401");`
	`this.result = this.gDateTime1.LessEqual(this.gDateTime2);`
	
	Result: `True`

GlobalDateTime.LessThan

Description	Indicates whether or not a global time value is less than another global time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.gDateTime2 = GlobalDateTime.ParseFromString("20100402235401");`
	`this.result = this.gDateTime1.LessThan(this.gDateTime2);`
	
	Result: `True`

GlobalDateTime.Equals

Description	Indicates whether or not a global date and time value is equal to another global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.gDateTime2 = GlobalDateTime.ParseFromString("20100402235401");`
	`this.result = this.gDateTime1.Equals(this.gDateTime2);`
	
	Result: `False`

GlobalDateTime.IsBetween

Description	Indicates whether or not a global date and time value is between two other specified global date and time values.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.gDateTime2 = GlobalDateTime.ParseFromString("20100330235401");`
	`this.gDateTime3 = GlobalDateTime.ParseFromString("20100402235401");`
	`this.result =`
	`this.gDateTime2.IsBetween(this.gDateTime1,this.gDateTime3);`
	
	Result: `True`

GlobalDateTime.GetDate

Description	Returns the date of a global date and time value.

Example	`this.gDateTime = GlobalDateTime.ParseFromString("20100215104352");`
	`this.result = this.gDateTime.GetDate();`
	Result: Date value `20100215`

GlobalDateTime.GetTime

Description	Returns the time of a global date and time value.

Example	`this.gDateTime = GlobalDateTime.ParseFromString("20100215104352");`
	`this.result = this.gDateTime.GetTime();`
	Result: Time value `104352`

GlobalDateTime.GetDay

Description	Returns the day of a global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.result = this.gDateTime.GetDay( );`
	Result: String value `15`

GlobalDateTime.GetMonth

Description	Returns the month of a global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.result = this.gDateTime.GetMonth( );`
	Result: String value `02`

GlobalDateTime.GetYear

Description	Returns the year of a global date and time value.
Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");` `this.result = this.gDateTime.GetYear( );`
Result	**Result:** String value `2010`

GlobalDateTime.GetWeek

Description	Returns the calendar week of a global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.result = this.gDateTime.GetWeek( );`
	Result: String value `07`

GlobalDateTime.GetHour

Description	Returns the hours of a global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.result = this.gDateTime.GetHour( );`
	
	Result: String value `10`.

GlobalDateTime.GetMinute

Description	Returns the minutes of a global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.result = this.gDateTime.GetMinute( );`
	
	Result: String value `43`

GlobalDateTime.GetSecond

Description	Returns the seconds of a global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.result = this.gDateTime.GetSecond( );`
	
	Result: String value `52`

GlobalDateTime.AddDuration

Description	Adds a duration to a global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.result =`
	`this.gDateTime.AddDuration(Duration.ParseFromString("P3DT1H"));`
	
	Result: Global date time value `20100218114352`

GlobalDateTime.SubtractDuration

Description	Subtracts a duration from a global date and time value.

Example	`this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`
	`this.result =`
	`this.gDateTime.SubtractDuration(Duration.ParseFromString("P3DT1H"));`
	
	Result: Global date time value `20100212094352`

GlobalDateTime.Delta

Description	Returns the delta duration between two global date and time values.

Example

```
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.gDateTime2 = GlobalDateTime.ParseFromString("20100218114352");
this.result = this.gDateTime1.Delta(this.gDateTime2);
```

Result: Duration value `P3DT1H`

GlobalDateTime.StartOfMonth

Description: Returns the start of a month related to a global date and time value.

Example

```
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime.StartOfMonth();
```

Result: Global date time value `20100201000000`

GlobalDateTime.EndOfMonth

Description: Returns the end of a month related to a global date and time value.

Example

```
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime.EndOfMonth();
```

Result: Global date time value `20100228235959`

GlobalDateTime.StartOfYear

Description: Returns the start of a year related to a global date and time value.

Example

```
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime.StartOfYear();
```

Result: Global date time value `20100101000000`

GlobalDateTime.EndOfYear

Description: Returns the end of a year related to a global date and time value.

Example

```
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime.EndOfMonth();
```

Result: Global date time value `20101231235959`

GlobalDateTime.ConvertToDate

Description: Converts a global date and time value into a date.

Example

```
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime.ConvertToDate();
```

Result: Date value `20100215`

GlobalDateTime.ConvertToDateTime

Description	Converts a global date and time value into a date and time value.

> **i Note**
>
> Use this function only when an SAP business object uses the `DateTime` data type in the `http://sap.com/xi/BASIS/Global` namespace. Do not use the `DateTime` data type in your own business object definitions.

Example

```
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime.ConvertToDateTime();
```

Result: Date time value `2010-02-15T10:43:52.0000000Z`

GlobalDateTime.ConvertToLocalDateTime

Description	Converts a global date and time value into a local date and time value. You can specify target time zone . If no time zone is specified, the function converts the global date and time value into the time zone of the user that is currently logged on.

Example

```
this.gDateTime = GlobalDateTime.ParseFromString("20080330120000");
this.result = this.gDateTime.ConvertToLocalDateTime();
```

Result: Local date time value in the time zone of the user.

```
this.result = this.gDateTime.ConvertToLocalDateTime("CET");
```

Result: Local date time value `20080330140000 CET`

GlobalDateTime.ConvertToLocalNormalisedDateTime

Description	Converts a global date and time value into a local normalized date and time value. You can specify a time zone. If no time zone is specified, the time zone of the user that is currently logged on is used.

Example

```
this.gDateTime = GlobalDateTime.ParseFromString("20080330120000");
this.result = this.gDateTime.ConvertToLocalNormalisedDateTime();
```

Result: Local normalized date time value with the time zone of the user.

```
this.result =
this.gDateTime.ConvertToLocalNormalisedDateTime("CET");
```

Result: Local normalized date time value `20080330140000 CET`

GlobalDateTime.ParseFromString

Description	Converts a string into a global date and time value. The function accepts both the internal format and the ISO format.

> **i Note**
>
> This function supports static calls only.

Example	`this.result = GlobalDateTime.ParseFromString("20100215104352");` **Result:** Global date time value `20100215104352`

7.2.5.1.10 Language Code (Reuse Library)

This SAP reuse library contains a function for language codes.

LanguageCode.ParseFromString

Description	Converts a string into a language code. Before the conversion the system checks if the string is a valid language code. The function accepts ISO codes only. **i Note** This function supports static calls only.
Example	`this.description.languageCode = LanguageCode.ParseFromString("EN");` **Result:** Language code with content “E”.

7.2.5.1.11 Local Date and Time (Reuse Library)

This SAP reuse library contains functions that deal with local date and time values.

LocalDateTime.ConvertTimeZone

Description	Converts a local date and time value from one time zone into another time zone.

Example	`this.localDateTime = LocalDateTime.ParseFromString("20100215104352 CET");`
	`this.result = this.LocalDateTime.ConvertTimeZone("UTC");`
	
	Result: Local date time value `20100215094352 UTC`

LocalDateTime.ConvertToGlobalDateTime

Description	Converts a local date and time value into a global date and time value.

Example	`this.localDateTime = LocalDateTime.ParseFromString("20080330120000 CET");`
	`this.result = this.localDateTime.ConvertToGlobalDateTime();`
	
	Result: Global date time value `20080330100000`

LocalDateTime.ParseFromString

Description	Converts a string into a local date and time value. The function accepts both the internal format and the ISO format. **i Note** This function supports static calls only.
Example	`this.result = LocalDateTime.ParseFromString("20100215104352 CET");` **Result:** Local date time value `20100215104352 CET`

7.2.5.1.12 Local Normalised Date and Time (Reuse Library)

This SAP reuse library contains a function that deals with local normalized date and time values.

LocalNormalisedDateTime.ConvertToGlobalDateTime

Description	Converts a local normalized date and time value into a global date and time value.

Example	`import ABSL;`
	`var global2;`
	`var locnorm;`
	`var global3;`
	`global2 = GlobalDateTime.ParseFromString("20080330120000");`
	`locnorm = global2.ConvertToLocalNormalisedDateTime();`
	`global3 = locnorm.ConvertToGlobalDateTime();`
	
	Result: Global date time value `20080330120000`

7.2.5.1.13 Number Range (Reuse Library)

Overview

This SAP reuse library contains functions that you can use to draw numbers. The drawn numbers are separated by identifiers that support different number ranges for different object types, for example document types. These identifiers are called number range objects.

NumberRange.DrawNumber

Description	Draws a new number; ideally the next number in the sequence of a given number range object. The first value in the range always equals to 1; and the last value is 999.999.999. In cases where the current value has reached the maximum value 999.999.999, the system raises a message, and the number range is reset to 0. Hence, the next drawn number is 1.

Example	`var recordNumber = NumberRange.DrawNumber("RECORD");`
	`var numberRangeObject : ID = "ORDER";`
	`var orderNumber = NumberRange.DrawNumber( numberRangeObject );`
	
	Result: The value of the number range object RECORD and ORDER is the next number in sequence.

NumberRange.GetNumber

Description	Returns the current value of the number range object. In cases where there are no earlier calls to this specific number range object, the value is 0.

Example	`var recordNumber = NumberRange.GetNumber("RECORD");`
	`var numberRangeObject : ID = "ORDER";`
	`var orderNumber = NumberRange.GetNumber( numberRangeObject );`
	
	Result: The current value of the number range objects RECORD and ORDER are returned.

NumberRange.ResetRange

Description	Resets the value of the number range object to 0. The next drawn number is 1..

Example	`NumberRange.ResetRange("RECORD");`
	`var numberRangeObject : ID = "ORDER";`
	`NumberRange.ResetRange( numberRangeObject );`
	
	Result: The value of the number range objects RECORD and ORDER is set to 0.

7.2.5.1.14 Numeric (Reuse Library)

This SAP reuse library contains functions that you can use for integers and decimals.

Numeric.IsNumeric

Description	Indicates whether a string contains a numeric value or not. > **Note** > This function supports static calls only.
Example	`this.result = Numeric.IsNumeric("1.5A");` **Result:** False

Numeric.ParseFromString

Description	Converts a string into a numeric value. Before the conversion the system checks if the string is a valid numeric value. The decimal point can be a dot as well as a comma depending on the locale. **i Note** This function supports static calls only.

Examples	`this.result = Numeric.ParseFromString("1.5");`
	Result: Numeric value `1.5`
	`this.result = Numeric.ParseFromString("1,5");`
	Result: Numeric value `1.5`

Numeric.RoundToString

Description	Converts a numeric value into a string by rounding it to a given number of decimal places. The string format depends on the locale of the current user. If you want to ensure that the dot is used as decimal mark independently of the locale, you need to provide the optional `RawRepresentation` parameter. This parameter is useful, for example, when the sting needs to be parsed again into a numeric value using the `Numeric.ParseFromString` function.

Example	`this.pi = 3.14159;`
	`this.result = this.pi.RoundToString( 2 );`
	Result: `"3.14"` or `"3,14"` (depending on the locale of the user)
	→ Tip
	If you want to ensure that the dot is used as decimal mark consistently, use the following statement:
	`this.result = this.pi.RoundToString( 2 , true);`

7.2.5.1.15 Numeric Character (Reuse Library)

This SAP reuse library contains functions for codes and identifiers with implementation type NUMC.

For information about the data types and their implementation type, see the Business Center at https://www.sme.sap.com under *SAP Business ByDesign* > *Community* > *Wiki* > *SAP Business ByDesign Studio* > *Public Solution Model*. You can find the information in the **Published Data Types List**.

NumericCharacter.ParseFromString

Description	Converts a string into a numeric character. Before the conversion the system checks if the string is a valid numeric character. **i Note** This function supports static calls only.
Example	`this.numcCode.content = Numeric.ParseFromString("05");` **Result:** Numeric character value `05`

7.2.5.1.16 Quantity Conversion (Reuse Library)

This SAP reuse library contains functions that you can use to compare quantities and to perform simple calculations of quantities. The functions allows to use different units of measure if they are convertible.

QuantityConversion.GreaterEquals

Description	Indicates whether or not a quantity is greater than or equal to another quantity.

Example	`this.quantity1.Content = 52;`
	`this.quantity1.unitCode = "EA";`
	`this.quantity2.Content = 52;`
	`this.quantity2.unitCode = "EA";`
	`this.result = this.quantity1.GreaterEquals(this.quantity2);`
	
	Result: `True`

QuantityConversion.GreaterThan

Description	Indicates whether or not a quantity is greater than another quantity.

Example	`this.quantity1.Content = 52;`
	`this.quantity1.unitCode = "EA";`
	`this.quantity2.Content = 52;`
	`this.quantity2.unitCode = "EA";`
	`this.result = this.quantity1.GreaterThan(this.quantity2);`
	
	Result: `False`

QuantityConversion.LessEquals

Description	Indicates whether or not a quantity is less than or equal to another quantity.

Example

```
this.quantity1.Content = 52;
this.quantity1.unitCode = "EA";
this.quantity2.Content = 52;
this.quantity2.unitCode = "EA";
this.result = this.quantity1.LessEquals(this.quantity2);
```

Result: `True`

QuantityConversion.LessThan

Description: Indicates whether or not a quantity is less than another quantity.

Example

```
this.quantity1.Content = 52;
this.quantity1.unitCode = "EA";
this.quantity2.Content = 52;
this.quantity2.unitCode = "EA";
this.result = this.quantity1.LessThan(this.quantity2);
```

Result: `False`

QuantityConversion.Equals

Description: Indicates whether a quantity is equal to another quantity or not.

Example

```
this.quantity1.Content = 52;
this.quantity1.unitCode = "EA";
this.quantity2.Content = 52;
this.quantity2.unitCode = "EA";
this.result = this.quantity1.Equals(this.quantity2);
```

Result: `True`

QuantityConversion.Add

Description: Adds two quantities. You can add quantities of different units of measure (UoM). If you do not specify a target UoM, the function returns the result in the UoM of the first quantity.

Example

```
this.quantity1.Content = 53;
this.quantity1.unitCode = "EA";
this.quantity2.Content = 52;
this.quantity2.unitCode = "EA";
this.result = this.quantity1.Add(this.quantity2);
```

Result: Quantity value `105 EA`

QuantityConversion.Subtract

Description: Subtracts a quantity from another quantity. You can substract quantities of different units of measure (UoM). If you do not specify a target UoM, the function returns the result in the UoM of the first quantity.

Example	```this.quantity1.Content = 53;
	this.quantity1.unitCode = "EA";
	this.quantity2.Content = 52;
	this.quantity2.unitCode = "EA";
	this.result = this.quantity1.Subtract(this.quantity2);```
	Result: Quantity value `1 EA`

QuantityConversion.Divide

Description	Divides a quantity by another quantity. The result is a decimal value.

Example	```this.quantity1.Content = 58;
	this.quantity1.unitCode = "EA";
	this.quantity2.Content = 37;
	this.quantity2.unitCode = "EA";
	this.result = this.quantity1.Divide(this.quantity2);```
	Result: Decimal value `1,56756756756756`

7.2.5.1.17 Time (Reuse Library)

This SAP reuse library contains the following functions that you can use, for example, to compare times.

Time.GreaterEquals

Description	Indicates whether or not a time value is greater than or equal to another time value.

Example	```this.time1 = Time.ParseFromString("120157");
	this.time2 = Time.ParseFromString("230000");
	this.result = this.time1.GreaterEquals(this.time2);```
	Result: `False`

Time.GreaterThan

Description	Indicates whether or not a time is greater than another value.

Example	```this.time1 = Time.ParseFromString("120157");
	this.time2 = Time.ParseFromString("230000");
	this.result = this.time1.GreaterThan(this.time2);```
	Result: `False`

Time.LessEquals

Description	Indicates whether or not a time value is less than or equal to another value.

Example	`this.time1 = Time.ParseFromString("120157");`
	`this.time2 = Time.ParseFromString("230000");`
	`this.result = this.time1.LessEquals(this.time2);`

	Result: `True`

Time.LessThan

Description	Indicates whether or not a time value is less than another value.

Example	`this.time1 = Time.ParseFromString("120157");`
	`this.time2 = Time.ParseFromString("230000");`
	`this.result = this.time1.LessThan(this.time2);`

	Result: `True`

Time.GetHour

Description	Returns the hour of a time value.
Example	`this.time = Time.ParseFromString("120157");` `this.result = this.time.GetHourl();`
	Result: String value `12`

Time.GetMinute

Description	Returns the minutes of a time value.
Example	`this.time = Time.ParseFromString("120157");` `this.result = this.time.GetMinute();`
	Result: String value `01`

Time.GetSecond

Description	Returns the seconds of a time value.
Example	`this.time = Time.ParseFromString("120157");` `this.result = this.time.GetSecond();`
	Result: String value `57`

Time.Delta

Description	Returns the duration delta between two specified time values.

Example	`this.time1 = Time.ParseFromString("120157");`
	`this.time2 = Time.ParseFromString("230000");`
	`this.result = this.time1.Delta(this.time2);`

	Result: Duration value `PT10H58M3S`

Time.ParseFromString

Description	Converts a string into a time value. The function accepts both the internal format and the ISO format. **i Note** This function supports static calls only.
Example	`this.result = Duration.ParseFromString("120157");` **Result:** Time value `12:01:57`

7.2.5.1.18 UUID (Reuse Library)

This reuse library contains functions for Universal Unique Identifiers (UUIDs).

UUID.ParseFromString

Description	Converts a string into a UUID. **i Note** This function supports static calls only.
Example	`this.result.content = UUID.ParseFromString("00163E01023602DD88AFE2F34FC4C3F3");` **Result:** UUID `00163E01023602DD88AFE2F34FC4C3F3`

UUID.Generate

Description	Generates a UUID. **i Note** This function supports static calls only.
Example	`this.result.content = UUID.Generate();` **Result:** A UUID that is different each time the function is called.

7.2.5.1.19 Sensitive Personal Data (Reuse Library)

This SAP reuse library contains functions for read access logging (RAL).

SensitivePersonalData.WriteDirectAcessLog

Description	Writes direct access logs when sensitive personal data is read and sent through one of the channels. Channels can be, for example, ABSL_MSG.
API Input Parameters	RALChannelCode : Channel Code to be passed RALLogItems : Ral Log for the fields ChannelContextDescription (OPTIONAL) : Additional description of the channel context ChannelContextDetails (OPTIONAL) : Any additional details regarding the channel context

Example	`var code : RALChannelCode;`
	`code = "ABSL_MSG";`
	`raise`
	`DemoMessage.Create("I",this.TOCUST.TaxNumber.GetFirst().TaxID.content);`
	`RALLogItem1 =`
	`this.TOCUST.TaxNumber.GetFirst().TaxID.content.GetRALLogItem();`
	`RALLogItemCollection.Collection.Add(RALLogItem1);`
	`SensitivePersonalData.WriteDirectAccessLog(code.GetDescription(),RALLogItemCollection);`

SensitivePersonalData.WritePreparedDataLog

Description	Writes prepared data logs when attachments with sensitive data are created using sensitive fields.
API Input Parameters	BusinessObjectNode : Node information for logging RALLogItemCollection : Ral log for the fields ChannelContextDescription (OPTIONAL) : Additional description of the channel context ChannelContextDetails (OPTIONAL) : Any additional details regarding the channel context

Example	`var myAttachmentFolder = this.SecAttachment.Create();`
	`varchannelContDesc = “Additional Description for channel context”;`
	`varchannelContDet = “Additional channel context details”;`
	`myAttachmentFolder.CreateFile( docType ,`
	`docName ,docAltName ,docDesc ,binaryObject ;`
	`RALLogItem1 =`
	`this.TOEMP.Common.GetFirst().Person.BirthPlaceName.GetRALLogItem();`
	`RALLogItemCollection.Collection.Add(RALLogItem1);`
	`SensitivePersonalData.WritePreparedDataLog(this.SecAttachment.Document.GetFirst().FileContent,RALLogItemCollection,channelContDesc,channelContDet);`

GetRALLogItem

Description	Creates an instance of RAL log item, that will be recorded in the Read Access Log. This context function can be used only for the fields marked as Sensitive Personal Data.

Example

```
var  code :  RALChannelCode;
code = "ABSL_MSG";
raise
DemoMessage.Create("I",this.TOCUST.TaxNumber.GetFirst().TaxID.conten
t);
RALLogItem1  =
this.TOCUST.TaxNumber.GetFirst().TaxID.content.GetRALLogItem();
RALLogItemCollection.Collection.Add(RALLogItem1);
SensitivePersonalData.WriteDirectAccessLog(code.GetDescription(),RAL
LogItemCollection);
```

GetChangeLogInfo

Description

Fetches the BO name, Node name, or Node ID when used with the ChangeHistoryEnabled node.

Use this function to delete the ChangeLogHistory attached to the respective node in deletion BADI absl.

Example

```
var  result :
DataDestructionBAdIDestroyCustomDataResultDataStructure;
```

```
var  ChangelogInfo :
DataDestructionBAdIDestroyCustomDataResultChangeDocumentNodeDetails;
var  ChangelogInfoCollection :collectionof
DataDestructionBAdIDestroyCustomDataResultChangeDocumentNodeDetails;
```

```
if(InputData..BusinessPartnerRole  == "Private Account")
{
```

```
  var  CustRoot  =
Customer .Retrieve(InputData.BusinessPartnerUUID);
  var CustomerID  = CustRoot.InternalID;
```

```
  var  PrivateAccountRoot = PrivateAccntCopy.Retrieve(CustomerID);
  raise  NjInfo.Create("I");
```

```
  var  time = Library::Context.GetCurrentUserTime();
  var  str = time.ToString();
```

```
  PrivateAccountRoot.Account_ID = "XXXX".Concatenate(str);
```

```
  ChangelogInfo = PrivateAccountRoot.GetChangeLogInfo();
  ChangelogInfoCollection.Add(ChangelogInfo);
  ChangelogInfo.Clear();
```

```
  result.ChangeDocumentNodeDetails = ChangelogInfoCollection;
   result.ErrorOccurredDuringDestroy = false;
```

```
    }
return result;
```

7.2.5.1.20 Web Service Utilities

This SAP reuse library contains functions related to Web services.

For more information, see Web Services Quick Guide [page 586].

WebServiceUtilities.ExecuteWebService

Description	Calls a REST or RSS/Atom mashup Web service. This reuse library focuses on relatively simple Web services that can be modeled as a mashup Web service. If you need to control the Web service call in more detail, you can use the `WebServiceUtilities.ExecuteRESTService` reuse function. For more information, see Using a Mashup Web Service in a Script File [page 563].

Example	`var result = WebServiceUtilities.ExecuteWebService(serviceID, parameters);`
	`var content = result.ResponseContent;`
	`var returnCode = result.ReturnCode;`

WebServiceUtilities.ExecuteRESTService

Description	Calls a REST Web service defined in an External Web Service Integration scenario. For more information, see Integrate an External Web Service Using REST [page 614]. This reuse library allows you to control the Web service call using a set of parameters. You can, for example, dynamically create the request body or use specific authentication mechanisms or header parameters. The following parameters need to be filled: • `CommunicationScenarioName`: Name of the communication scenario (`.csd`) defined in the studio • `ServiceName`: Name of the External Web Service Integration scenario (`.wsid`) defined in the studio • `HttpMethod`: Function supports the following HTTP methods: 'GET', 'HEAD','PATCH', 'POST', 'PUT', 'DELETE' • `ContentType`: Specifies the character encoding (for example, text/html) of the request • `ResourceName`: Resource name of the rest service that is concatenated to the defined URL in the service of the communication arrangement • `URLParam`: Additional URL parameters that are concatenated to the defined URL in the service of the communication arrangement • `HeaderParameters`: Header parameters of the request • `Body`: Body of the request. Body has to fit with defined content type. • `Cookies`: Cookies that are to be set in the HTTP request. After executing the service, the function fills the following parameters in the result structure: • `ReturnCode`: Return code of the REST call, for example, `200`for success • `Content`: Content of the response • `HeaderParameters`: Header parameters of the response • `ContentType`: Content type of the response • `Cookies`: Cookies that are returned from the HTTP response

Example

```
var headerParams : collectionof NameAndValue;
var headerParam : NameAndValue;
var urlParams : collectionof NameAndValue;
var urlParam : NameAndValue;
headerParam.Name = "Accept-Encoding";
headerParam.Value = "gzip,deflate,sdch";
headerParams.add(headerParam);

headerParam.Name = "Accept-Language";
headerParam.Value = "de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4";
headerParams.add(headerParam);

urlParam.Name = "listZipCodeList";
urlParam.Value = "20910";
urlParams.add(urlParam);

var result =
WebServiceUtilities.ExecuteRESTService(communicationScenarioName,
serviceName,"GET", "ext/plain", "ndfdXMLclient.php", urlParam,
headerParams, "");
//You can obtain the response cookies from the result as shown below
varSetCookies = result.Cookies.
//You can set cookies in the HTTP request as shown below
varresult =
WebServiceUtilities.ExecuteRESTService(communicationScenarioName,service
Name,"POST", "ext/plain", "ndfdXMLclient.php", urlParam, headerParams,
"", SetCookies);
```

For more information, see Example: Consume a REST Service [page 616].

WebServiceUtilities.EncryptWithHashBasedMessageAuthenticationCode

Description

Encrypts a message with a keyed-hash message authentication code.

The following parameters need to be filled:

- `Scenario`: Communication scenario name
- `Algorithm`: Function supports the following hash algorithms: `'SHA256'`, `'SHA512'`
- `Key`: Placeholder for secret cryptographic key
- `Message`: Message that is encrypted with the specified key and hash algorithm

Example

```
var hmac =
WebServiceUtilities.EncryptWithHashBasedMessageAuthenticationCode(Scenar
io, "SHA256", Root.SecretKey, message);
```

7.2.5.2 SAP Reuse Libraries for Business Areas

7.2.5.2.1 SAP Reuse Libraries for Business Areas Reference

The SAP reuse libraries for business areas provide additional, very specific services to the scripting language that are relevant for functions in the business areas of the cloud solution from SAP, for example, in the Financial Management business area. These services are all based on entities provided by SAP.

> **Note**
>
> To use an SAP reuse library for a specific business areas, you need to import the namespace of the relevant business area with the *import* keyword, for example, `import AP.FO.ProductDataMaintenance.Global;`. For more information about the *import* keyword, see Syntax for Implementation of Business Logic [page 235].

List of SAP Reuse Libraries for Business Areas

- Accounting Coding Block Assignment Utilities [page 309]
  This SAP reuse library contains a service that is based on the `AccountingCodingBlockDistribution` dependent object provided by SAP.
- Address Snapshot Utilities [page 310]
  This SAP reuse library contains a service that is based on the `Address Snapshot` business object provided by SAP.
- Date Time Utilities [page 310]
  This SAP reuse library contains a service that is based on advanced functions for date and time processing.
- Financials Process Utilities [page 312]
  This SAP reuse library contains a service that is relevant for the Financial Management business area.
- Financials Set of Books Utilities [page 313] (relevant for SAP Business ByDesign only)
  This SAP reuse library contains a service that is based on the `Set of Books` business object provided by SAP.
- Material Utilities [page 314]
  This SAP reuse library contains a service that is based on the `Material` business object provided by SAP.
- Business Partner Utilities [page 316]
  This SAP reuse library contains a service that is based on the `Business Partner` business object provided by SAP.
- Price Component Utilities [page 317]
  This SAP reuse library contains a service that is based on the `PriceAndTaxCalculation` dependent object provided by SAP.
- Due Clearing Utilities [page 317]
  This SAP reuse library contains a service that is based on the `DueClearing` business object provided by SAP.

7.2.5.2.2 Accounting Coding Block Assignment Utilities

To use this SAP reuse library, you need to import the `AP.IS.CodingBlock` namespace.

AccountingCodingBlockAssignmentUtilities.ToMessage

Description	Converts the `AccountingCodingBlockAssignment` subnode of the `AccountingCodingBlockDistribution` dependent object into the `AccountingCodingBlockAssignment` global data type.
Example	`FormData.ItemAccountingCodingBlockAssignment = this.ToItemAccountingCodingBlockAssignment.ToMessage( );` In this example, `FormData.ItemAccountingCodingBlockAssignment` is the`AccountingCodingBlockAssignment` global data type. `this.ToItemAccountingCodingBlockAssignment` refers to the `AccountingCodingBlockAssignment` subnode of the `AccountingCodingBlockDistribution` dependent object. **Result:** Accounting coding block assignment.

7.2.5.2.3 Address Snapshot Utilities

To use this SAP reuse library, you need to import the `AP.FO.Address.Global` namespace.

AddressSnapshotUtilities.ToForm

Description	Converts an address into a format that can be used in a print form. The data stored in an instance of the `Address Snapshot` business object is converted into the `FormAddress` global data type.

Example	`this.SenderCountry = "US";`
	`this.MaximumOfLines = 12;`
	`this.UseStreetAddress = true;`
	`FormData.PurchaseOrder.SellerPartyAddress =`
	`this.SellerParty.AddressSnapshot.ToForm( this.SenderCountry,`
	`this.MaximumOfLines, this.UseStreetAddress );`
	
	In this example, `AddressSnapshot` refers to the `AddressSnapshot` business object.`FormData.PurchaseOrder.SellerPartyAddress` is the `FormAddress` global data type .
	
	Result: Form address, for example, FormData.PurchaseOrder.SellerPartyAddress.FormattedAddressDescription = "AMS / 166 South High Street / Akron OH 44308 / US"

7.2.5.2.4 Date Time Utilities

This SAP reuse library contains a service that is based on advanced functions for date and time processing. To use this library, you need to import the `AP.IS.DateAndTime.Global` namespace.

DateTimeUtilities.GetNextActiveTimePoint

Description	Returns the next active time value based on the calendar, direction (+ or -) and the date and time value passed. **i Note** If you do not provide a calendar, the default calendar is used.

Example	`this.CurrentDateTime.Date = Date.ParseFromString("20080330")`
	`this.CurrentDateTime.Time = Time.ParseFromString("101010");`
	`this.Direction = "+";`
	`this.CalCode.content = "11";`
	`this.NextDateTime =`
	`DateTimeUtilities.GetNextActiveTimePoint(this.CurrentDateTime,`
	`this.Direction, this.CalCode);`
	
	Result: Next date value: `31.03.2008`, next time value: `000000`

DateTimeUtilities.MoveTimePoint

Description	The MoveTimePoint operation moves a time point by a specified duration either forwards or backwards on a given calendar. The response structure returns the time point considering working or non-working days when a calendar is used for calculation. If the operation is used without a calendar, the default (gregorian) calendar is used. **i Note** If you do not provide a calendar, the default (gregorian) calendar is used. CalendarCode: ID of the working day calendar. CalendarUUID: UUID of a calendar schema with operating hours. CalendarUUID overrules the CalendarCode if both provided. Direction: plus (+) or minus (-). Duration: Required in ISO format e.g. "P2D" = 2 days, "PT48H" = 48 hours.

Example

```
this.CurrentDateTime.Date = Date.ParseFromString("20140908010000")
this.CurrentDateTime.TypeCode = "5"; (5 is Local Normalised Date
Time)
this.Duration = ABSL:Duration.ParseFromString("PT48H");
this.Direction = "+";
this.CalCode.content = "11";
this.CalUUID.content =
ABSL:UUID.ParesFromString("00000000000102DCAE83346B4F5F1B49";
this.NextDateTime =
DateTimeUtilities.MoveTimePoint(this.CurrentDateTime,
this.Duration, this.Direction, this.CalCode, this.CalUUID);
```

Result: Next date value: `20140916130000`

Result Explanation: Calendar Schema: Monday to Friday 09:00 to 17:00 based on EST, EST time zone difference to UTC: +4 hours(based on calculation day 08.09.2014) Calendar schema on UTC: Monday to Friday 13:00 to 21:00

Start time point is 08.09.2014 01:00:00 (UTC) which is outside the operating hours, so it starts at next active time point: 08.09.2014 13:00:00

Every working day has 8 operating hours so 48 hours correspond to 6 working days. The calculation ends on 15.09.2014 21:00:00 so next active time point is 16.09.2014 13:00:00.

7.2.5.2.5 Financials Process Utilities

This reuse library contains a service that is relevant for the Financial Management business area. To use this library, you need to import the `AP.FO.CompanyFinancialsProcessControl.Global` namespace.

FinancialsProcessUtilities.GetProcessLockStatus

Description

Returns the process lock status for a given accounting transaction date, company and business transaction type group. Please note that on the user interface of SAP Business ByDesign, process lock is called “Process Control” and accounting transaction date is called “Posting Date” or “Proposed Posting Date” .

For information about the process lock, see the SAP Business ByDesign Library under *Business Areas* > *Financial Management* > *Financial and Management Accounting* > *General Leger* > *Master Data* > *Companies View* > *Quick Guide for Companies (General Ledger)* > *Set the Process Control*.

> **i Note**
>
> This service supports static calls only.

Example

```
this.BTTGroup = "02";// Customer Invoicing
this.TransactionDate = Date.ParseFromString("20110623");
this.BlockingStatus =
FinancialsProcessUtilities.ProcessLock( this.TransactionDate,
this.BTTGroup, this.ToCompany );
```

In this example, `this.ToCompany` refers to the `Company` business object provided by SAP.

Result: Process lock status code, for example, `1` for status "not blocked".

7.2.5.2.6 Financials Set of Books Utilities (SAP Business ByDesign Only)

This SAP reuse library contains a service that is based on the `Set of Books` business object provided by SAP. To use this library, you need to import the `AP.FinancialAccounting.Global` namespace.

SetofBooksUtilities.GetFiscalYearStartEndDate

Description	Returns the start date and the end date of the fiscal year for a given company, set of books, and fiscal year.

Example

```
import ABSL;
import AP.Common.GDT as common;
import AP.FinancialAccounting.Global;
import AP.FO.MOM.Global;
var FYDates : DatePeriod;
var comp : Company;
var fyID : FiscalYearID;
var ext: Fisc;
var tempDate : common:Date;
var year : Year;
if(!this.CompanyId.IsInitial())
{
  if(!this.sobid.content.IsInitial())
  {
    comp = Company.Retrieve(this.CompanyId);
    var temp = "2012";
    fyID = NumericCharacter.ParseFromString(temp);
      if(comp.IsSet())
      {
        FYDates =
SetofBooksUtilities.GetFiscalYearStartEndDate(this.sobid, fyID,
comp);
        this.FYStartDate = FYDates.StartDate;
        this.FYEndDate = FYDates.EndDate;
      {
    {
  {
```

Result:

Start date of the fiscal year, for example,`01012012` for year 2012

End date of the fiscal year, for example, `31122012` for year 2012

7.2.5.2.7 Material Utilities

This SAP reuse library contains a service that is based on the `Material` business object provided by SAP. To use this library, you need to import the `AP.FO.ProductDataMaintenance.Global` namespace.

MaterialUtilities.QuantityConversion

Description	Converts a quantity of a material or product into a different unit of measure or quantity specification.
Example	```this.quantity_in.content = 1; this.quantity_in.unitCode = "XPX"; this.unit_out = "EA"; this.quantity_out = this.toMaterial.QuantityConversion(this.quantity_in, this.unit_out);``` In this example, `toMaterial` refers to the `Material` business object. **Result:** Quantity value, for example, `20 EA`

7.2.5.2.8 Output Management Utilities

This SAP reuse library contains a service that is based on output management functionality. To use this reuse library, you need to import the following namespaces:

- `AP.Common.GDT`
- `BASIS.Global`
- `DocumentServices.Global`

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

OutputManagementUtilities.GetPDF

Description	Provides business object data in a portable document format (PDF) document.

Example	`//variables`
	`var FormTemplateLanguage  = "E";`
	`var PDF : BinaryObject;`
	`var FormTemplateCode : OutputRequestFormTemplateCode;`
	`FormTemplateCode.content = "ZMEI2TXR_P3JE6"; //Code is Form Template Header Code`
	
	`// Reuse Service Call`
	`PDF =`
	`OutputManagementUtilities.GetPDF(this,FormTemplateCode,FormTemplateLanguage);`
	
	Result: A PDF, which can be used further for storage, for example, as a business object attachment.

OutputManagementUtilities.CreatePrintQueueItem

Description	Enables you to send custom PDFs to the print queue.

Example	`import ABSL;`
	`import AP.Common.GDT;`
	`import BASIS.Global;`
	`import DocumentServices.Global;`
	`varuserAccountID:UserAccountID;`
	`userAccountID.content =“Enter USER ID”;`
	`var binaryContent:BinaryObject;`
	`binaryContent.content = "255044462D312E360D25E2E3CFD30D0A31312030206F626A0A3C3C2F46696C7465722F466C6174654465636F64652F466972737420392F4C656E677468203136";`
	`//Input for CreatePrintQueueItem:`
	`//[Login User ID, Document Name, No. of Copies, PDF Binary Content, Queue ID]`
	`this.PrintUUID =`
	`OutputManagementUtilities.CreatePrintQueueItem(userAccountID,“<Document Name>”,1,binaryContent,"DEFAULT");`

OutputManagementUtilities.CreatePrintQueueItemWithFormTemplate

Description	Enables you to create print queue instances based on the form template code, country code, language fields as inputs to fetch the corresponding form template. The XML binary content along with the form template is used to generate the actual PDF content.

Example

```
import ABSL;
import AP.Common.GDT;
import BASIS.Global;
import DocumentServices.Global;
varuserAccountID:UserAccountID;
userAccountID.content =“<Enter USER ID>”;
var binaryContent:BinaryObject;
var lang:DataType::LanguageCode;
var country:CountryCode;
var formTemplateCode:OutputRequestFormTemplateCode;
var companyID:OrganisationalCentreID;
var orgUnit:OrganisationalCentreID;
var channelType:CommunicationMediumTypeCode;
formTemplateCode = "C0001";
lang = "EN";
country ="US";
channelType ="PRT";
companyID ="MC10000";
orgUnit ="MC42100";
binaryContent.content = "255044462D312E360D25E2E3CFD30D0A31312030206F626
A0A3C3C2F46696C7465722F466C6174654465636F64652F466972737420392F4C656E677
468203136";
//Input for CreatePrintQueueItemWithFormTemplate:
//[Login User ID, Document Name, Queue ID, No. of Copies, PDF Binary
Content, FormTemplateCode, Language, Country Code, CompanyID, Org. Unit
ID, Channel Type]
this.PrintUUID =
OutputManagementUtilities.CreatePrintQueueItemWithFormTemplate(userAccou
ntID,"<Document Name>","DEFAULT",
1,binaryContent,formTemplateCode,lang,country,companyID,orgUnit,channelT
ype);
```

7.2.5.2.9 Business Partner Utilities

To use this SAP reuse library, you need to import the `AP.FO.BusinessPartner.Global` namespace

BusinessPartnerUtilities.IsBusinessPartnerMerge

Description

Returns true if a Business Partner Merge Run is executed in the current transaction.

Developers can decide to use this flag in their custom development depending on the scenario. For example, developers can skip certain determinations during the merge process.

Example

```
var isMergeContext:Indicator;
isMergeContext = BusinessPartnerUtilities.IsBusinessPartnerMerge( );
```

Result:

True - Transaction is running in merge context.

False - Transaction is not running in merge context

.

BusinessPartnerUtilities.IsBusinessPartnerMergeEnhancedRealignment

Description	Returns true if the enhanced re-alignment phase of the Business Partner Merge Run is executed in the current transaction. Developers can decide to use this flag in their custom development depending on the scenario. For example, developers can skip certain determinations during the merge process.

Example	`var isEnhancedRealignmentMergeContext:Indicator;`
	`isEnhancedRealignmentMergeContext = BusinessPartnerUtilities.;`
	`IsBusinessPartnerMergeEnhancedRealignment( );`
	
	Result:
	
	True - Transaction is running in merge enhanced re-alignment context.
	
	False - Transaction is not running in merge enhanced re-alignment context.

7.2.5.2.10 Price Component Utilities

This SAP reuse library contains a service that is based on the `PriceAndTaxCalculation` dependent object provided by SAP. To use this library, you need to import the `AP.FO.PriceAndTax.Global` namespace.

PriceComponentUtilities.ToMessage

Description	Converts the`PriceAndTaxCalculation` dependent object into the `FormPriceComponent` global data type.

Example	`FormData.PriceComponent =`
	`this.PriceAndTaxCalculation.PriceComponent.ToMessage( );`
	
	In this example, `FormData.PriceComponent` is the `FormPriceComponent` global data type `.this.PriceAndTaxCalculation.PriceComponent` refers to the `PriceAndTaxCalculation` dependent object.
	
	Result: Form price component.

7.2.5.2.11 Due Clearing Utilities

This SAP reuse library contains a service that is based on the `DueClearing` business object provided by SAP. To use this library, you need to import the `AP.DueItemManagement.Global` namespace.

> **i Note**
>
> This reuse library is valid for Business ByDesign only.

DueClearingUtilities.ClearOpenItems

Description	Clears open items provided as input to the utility. **Input parameters:** OpenItemsList Datatype Name: `DueClearingClearOpenItemsElements` Elements: • `PostingDate` Datatype: Date • `DocumentDate` Datatype: Date • `ItemDetails` Datatype: `DueClearingOpenItemDetailsList` (List of open items) Cardinality (1..*) Elements of `DueClearingOpenItemDetailsList`: • `TRPItemUUID` Datatype: UUID • `TransactionCurrencyCode` Datatype: CurrencyCode • `AppliedNetAmount` Datatype: Amount • `CashDiscountAmount` Datatype: Amount • `DeductionsList` Datatype: `DueClearingOpenItemDetailsDeductionsList` (Details of deductions for open item) (Cardinality 0..*) Elements of `DueClearingOpenItemDetailsDeductionsList`: • `DeductionAmount` Datatype: Amount • `TransactionCurrencyCode` Datatype: CurrencyCode • `PaymentDifferenceReasonCode` Datatype: `PaymentDifferenceReasonCode` **Output parameters:** Result Datatype: `DueClearingClearOpenItemsResultElements` Elements: `CreatedClearingUUID` - Consists of the resultant `DueClearing` UUID

Example

```
import ABSL;
import AP.DueItemManagement.Global;
import AP.Common.GDT;
```

```
// Variables
var result;
var OpenItems: DueClearingClearOpenItemsElements;
var PostingDate;
var DocumentDate;
```

```
var listOpenItems: collectionof DueClearingClearOpenItemsElements;
var ItemDetails: DueClearingOpenItemDetailsList;
```

```
var TRPItemStringUUID;
var TRPItemUUID: UUID;
var ItemTransactionCurrencyCode;
var AppliedNetAmount;
var CashDiscountAmount;
```

```
var DeductionsList;
var DeductionAmount;
var DedTransactionCurrencyCode;
var PaymentDifferenceReasonCode;
```

```
PostingDate = Library::Date.ParseFromString("20160601");
DocumentDate = Library::Date.ParseFromString("20160601");
```

```
OpenItems.PostingDate = PostingDate;
OpenItems.DocumentDate = DocumentDate;
```

```
//Fetch Items for your usecase by querying BO Trade Receivables
Payables Register - Item Overview Node - QueryByElements
```

```
var TRPItemUUID1 =
Library::UUID.ParseFromString("00163E04ECFD1ED5B5AA17AF9EAB280F");
```

```
//Assign Item Details
ItemDetails.TRPItemUUID.content = TRPItemUUID1;
ItemDetails.TransactionCurrencyCode = "USD";
ItemDetails.AppliedNetAmount.content = 20;
OpenItems.ItemDetails.Add(ItemDetails);
```

```
//Fetch Items for your usecase by querying BO Trade Receivables
Payables Register - Item Overview Node - QueryByElements
```

```
var TRPItemUUID2 =
Library::UUID.ParseFromString("00163E04ECFD1ED5B5AA26174E186835");
```

```
//Assign Item Details
ItemDetails.TRPItemUUID.content = TRPItemUUID2;
ItemDetails.TransactionCurrencyCode = "USD";
ItemDetails.AppliedNetAmount.content = 20;
OpenItems.ItemDetails.Add(ItemDetails);
```

```
listOpenItems.Add(OpenItems);
```

```
result =
Library::DueClearingUtilities.ClearOpenItems(listOpenItems);
```

7.2.6 SAP Built-In Functions

7.2.6.1 SAP Built-In Functions Reference

The SAP built-in functions of the scripting language extend the scope of the syntax for the implementation of your business logic in a similar way as the basic reuse libraries. However, in contrast to the reuse libraries, you do not need to import a namespace to use the built-in functions.

List of SAP Built-In Functions

The built-in functions are listed in the following:

- Generic Built-In Functions [page 320]
- Built-In Functions for Codes [page 321]
- Built-In Functions for Strings [page 322]
- Built-In Functions for Numeric Characters [page 328]
- Built-In Functions for Identifiers [page 329]
- Tracing Built-In Functions [page 643]

7.2.6.2 Generic Built-In Functions

These functions bind to more than one data type.

Function	Binds to Data Types	Description
Clear	All data types except for Boolean data types.	Clears a value.
IsInitial	All data types except for Boolean data types.	Indicates whether a value is initial or not. Returns a Boolean value.

Function	Binds to Data Types	Description
ToString	• All base types except for string data types • All non-base types that are not structured • Amount (http://sap.com/xi/AP/Common/GDT) • Quantity (http://sap.com/xi/AP/Common/GDT) • LOCAL_DateTime (http://sap.com/xi/BASIS/Global) • LOCALNORMALISED_DateTime (http://sap.com/xi/BASIS/Global)	Converts a value into a string. For binary values UTF-8 encoding is used. Invalid byte sequences are replaced by the replacement character (U+FFFD). The replacement character is shown as a black diamond with a white question mark.

Example:

```
if (!this.numericValue.IsInitial()) {
  this.string = this.numericValue.ToString();
  this.numericValue.Clear();
 }
```

7.2.6.3 Built-In Functions for Codes

You can use these functions to read the description of numeric code data types or character-based code data types. If you use a structured data type, the method binds to the structure but not to the content element.

GetDescription

Description	Returns the description of a code value in the language of the user. For some data types, for example, if you use the `RegionCode` data type, you also need to provide the `ListID` element.

Examples	`var codeDescriptionInUserLanguage   = code.GetDescription();`
	`var codeDescriptionFrenchRegionCode =`
	`regionCode.GetDescription("FR");`

GetDescriptionInOtherLanguage

Description	Returns the description of a code value in a specified language. For some data types, for example, if you use the `RegionCode` data type, you also need to provide the `ListID` element.

Example	`var codeDescriptionInEnglish        =`
	`code.GetDescriptionInOtherLanguage(LanguageCode.ParseFromString("EN"`
	`));`

GetAllValuesAndDescriptions

Description	Returns a collection of values and descriptions for a code data type in the language of the user. For some data types, for example, if you use the `RegionCode` data type, you also need to provide the `ListID` element.
Examples	`var allCodeValuesDescriptionsInUserLanguage = code.GetAllValuesAndDescriptions(); var allFrenchRegionCodeValuesAndDescriptions = regionCode.GetAllValuesAndDescriptions("FR");`

GetAllValuesAndDescriptionsInOtherLanguage

Description	Returns a collection of values and descriptions for a code data type in a specified language. For some data types, for example, if you use the `RegionCode` data type, you also need to provide the `ListID` element.
Examples	`var allCodeValuesAndDescriptionsInEnglish = code.GetAllValuesAndDescriptionsInOtherLanguage(LanguageCode.ParseFromString("EN")); var allFrenchRegionCodeValuesAndDescriptionsInEnglish = regionCode.GetAllValuesAndDescriptionsInOtherLanguage(LanguageCode.ParseFromString("EN"),"FR");`

7.2.6.4 Built-In Functions for Strings

These functions contain description and processing functions for character strings.

Concatenate

Description	Concatenates two strings.
Example	`this.string = "aaabbbccc"; this.result = this.string.Concatenate("test");` **Result:** String value `aaabbbccctest`

Contains

Description	Indicates whether or not a string contains a substring. If the source string is empty, the result is `false` and the system raises an information message. If the substring is empty, the result is `false` and the system raises an error message.
Example	`this.string = "aaabbbccc"; this.result = this.string.Contains("bbb");` **Result:** `True`

EndsWith

Description	Indicates whether a string ends with a substring or not. If the source string is empty, the result is `false` and the system raises an information message. If the substring is empty, the result is `false` and the system raises an error message.

Example	`this.string = "aaabbbccc";`
	`this.result = this.string.EndsWith("ccc);`
	Result: `True`

Find

Description	Searches from left to right and returns the position of a substring in a string. Optionally, you can define a search area by specifying the start position only or by specifying the start position as well as the length of the search area. If no substring is found, the function returns the value "-1". The function returns the value “-1” and raises a message if the system encounters one of the following issues: • The substring is empty (error message) • The start position is not within the length of the string (error message) • The length succeeds the length of the string (error message) • The source string is empty (information message)

Example	`this.string = "aaabbbcccbbb";`
	`this.result = this.string.Find("bbb");`
	Result: Integer value `3`
	Specifying the start position of the search area:
	`this.string = "aaabbbcccbbb";`
	`this.result = this.string.Find("bbb", 5);`
	Result: Integer value `9`
	Specifying the start position and lengh of the search area:
	`this.string = "aaabbbcccbbb";`
	`this.result = this.string.Find("bbb", 5, 3);`
	Result: Integer value `-1`

FindLast

Description	Searches from right to left and returns the position of a substring in a string. Optionally, you can define a search area by specifying the start position only or by specifying the start position and the length of the search area. If no substring is found, the function returns the value "-1". The function returns the value “-1” and raises a message if the system encounters one of the following issues: • The substring is empty (error message) • The start position is not within the length of the string (error message) • The length succeeds the length of the string (error message) • The source string is empty (information message)

Example	`this.string = "aaabbbcccbbb";`
	`this.result = this.string.FindLast("bbb");`
	
	Result: Integer value `9`
	
	Specifying the start position of the search area:
	
	`this.string = "aaabbbcccbbb";`
	`this.result = this.string.FindLast("bbb", 8);`
	
	Result: Integer value `3`
	
	Specifying start position and length of the search area:
	
	`this.string = "aaabbbcccbbb";`
	`this.result = this.string.FindLast("bbb", 8, 5);`
	
	Result: Integer value `3`

FindLastRegEx

Description	Searches from right to left and returns the position of a regular expression pattern in a string. Optionally, you can define a search area by specifying the start position only or by specifying the start position and the length of the search area. If no substring is found, the function returns the value "-1". The function returns the value “-1” and raises a message if the system encounters one of the following issues: • The substring is empty (error message) • The start position is not within the length of the string (error message) • The length succeeds the length of the string (error message) • The source string is empty (information message) For information about the operators that you can use in a regular expression pattern, see the *ABAP Keyword Documentation*.

Example

```
this.string = "aaa bbb ccc bbb ccc";
this.result = this.string.FindLastRegex("b\\s+");
```

Result: Integer value `14`

Specifying the start position of the search area:

```
this.string = "aaa bbb ccc bbb ccc";
this.result = this.string.FindLastRegex("b\\s+",8);
```

Result: Integer value `6`

Specifying start position and length of the search area:

```
this.string = "aaa bbb ccc bbb ccc";
this.result = this.string.FindLastRegex("b\\s+", 16, 2);
```

Result: Integer value `14`

FindRegEx

Description

Searches from left to right and returns the position of a regular expression pattern in a string. Optionally, you can define a search area by specifying the start position only or by specifying the start position and the length of the search area. If no substring is found, the function returns the value "-1".

The function returns the value “-1” and raises a message if the system encounters one of the following issues:

- The substring is empty (error message)
- The start position is not within the length of the string (error message)
- The length succeeds the length of the string (error message)
- The source string is empty (information message)

For information about the operators that you can use in a regular expression pattern, see the *ABAP Keyword Documentation*.

Example

```
this.string = "aaa bbb ccc bbb ccc";
this.result = this.string.FindRegex("b\\s+");
```

Result: Integer value `6`

Specifying the start position of the search area:

```
this.string = "aaa bbb ccc bbb ccc";
this.result = this.string.FindRegex("b\\s+", 8);
```

Result: Integer value `14`

Specifying start position and length of the search area:

```
this.string = "aaa bbb ccc bbb ccc";
this.result = this.string.FindRegex("b\\s+", 8, 5);
```

Result: Integer value `–1`

Length

Description	Returns the length of a string.

Example	`this.string = "test";`
	`this.result = this.string.Length();`
	
	Result: Integer value `4`

Matches

Description	Indicates whether or not the source string matches the regular expression pattern. The function raises a message and returns the result `false`, if the system encounters one of the following issues: • The regular expression pattern is empty (error message) • The regular expression pattern is invalid (error message) • The source string is empty (information message)

Example	`this.string = "entitlement";`
	`this.RegEx = "entitle.ent";`
	`this.result = this.string.Matches(this.RegEx);`
	
	Result: `True`

Replace

Description	Replaces all occurrences of a substring in a string with another substring. If the substring is empty, the function returns the source string. If the source string is empty, the function returns an empty string.

Example	`this.string = "aaabbbccc";`
	`this.result = this.string.Replace("bbb", "test");`
	
	Result: String value `aaatestccc`

ReplaceRegEx

Description	Returns all locations in a string with a substring that match a regular expression pattern. Optionally, you can define a search area by specifying the start position only or by specifying the start position and the length of the search area. If the substring is empty or the regular expression pattern is invalid, the function returns the source string and raises an error message. If the source string is empty, the function returns an empty string and raises an information message. For information about the operators that you can use in a regular expression pattern, see the *ABAP Keyword Documentation*.

Example	`this.string = "aaa bbb ccc bbb";`
	`this.result = this.string.ReplaceRegex("b\\s+","test");`
	
	Result: String value `aaa bbtestccc bbb`

StartsWith

Description	Indicates whether or not a string starts with a substring. If the source string is empty, the result is `false` and the system raises an information message. If the substring is empty, the result is `false` and the system raises an error message.

Example	`this.string = "aaabbbccc";`
	`this.result = this.string.StartsWith("ccc");`
	Result: `False`

Substring

Description	Returns a substring of a string starting at a specified position. Optionally, you can define the length of the substring. The function raises a message and returns an empty string, if the system encounters one of the following issues: • Either the start position or the sum of start position and length of the substring is not within the length of the string (error message) • The length of the substring succeeds the length of the string (error message) • The source string is empty (information message) • The start position or the length of the substring is a negative number (error message)

Example	• Substring of a string starting at a specified position:
	`this.string = "aaabbbccc";`
	`this.result = this.string.Substring(3);`
	Result: String value `bbbccc`
	• Defined length of the substring:
	`this.string = "aaabbbccc";`
	`this.result = this.string.Substring(4,3);`
	Result: String value `bbc`

ToLowerCase

Description	Converts a string into lower case. If the source string is empty, the system returns an empty string and raises an information message.

Example	`this.string = "AAAbbbCCC";`
	`this.result = this.string.ToLowerCase();`
	Result: String value `aaabbbccc`

ToUpperCase

Description	Converts a string into upper case. If the source string is empty, the system returns an empty string and raises an information message.

Example

```
this.string = "AAAbbbCCC";
this.result = this.string.ToUpperCase();
```

Result: String value `AAABBBCCC`

Trim

Description

Deletes leading and trailing characters. It is not relevant for the result, in which order you pass the characters to the system. If you do not specify a character, the system removes all blank characters.

If the source string is empty, the system raises an information message and returns an empty string.

Example

```
this.string = " aaabbbccc ";
this.result = this.string.Trim();
```

Result: String value `aaabbbccc`

TrimLeft

Description

Deletes leading characters. It is not relevant for the result, in which order you pass the characters to the system. If you do not specify a character, the system removes all blank characters.

If the source string is empty, the system returns an empty string and raises an information message.

Example

```
this.string = "xyz xyz abcd";
this.result = this.string.TrimLeft("zyx");
```

Result: String value `xyz abcd`

TrimRight

Description

Deletes trailing characters. It is not relevant for the result, in which order you pass the characters to the system. If you do not specify a character, the system removes all blank characters.

If the source string is empty, the system returns an empty string and raises an information message.

Example

```
this.string = "xyz xyz abcd";
this.result = this.string.TrimRight("bd ca");
```

Result: String value `xyz xyz`

7.2.6.5 Built-In Functions for Numeric Characters

This function binds to code and identifier data types with the `NUMC` implementation type.

You can use the *Repository Explorer* tool window to explore the SAP data types that are released with the public solution model (PSM) and their implementation type. For more information, see Repository Explorer [page 49].

ToNumeric

Description

Converts a numeric character into a numeric value.

Example	`var numeric = numcCode.ToNumeric();`

7.2.6.6 Built-In Functions for Identifiers

You can use these functions to remove leading zeros from or add leading zeros to numeric identifier data types and character-based identifier data types.

> **i Note**
>
> As a prerequisite for this function, the alpha-conversion indicator for the data type must be set.

AddLeadingZeros

Description	Adds leading zeros to the identifier.
Example	`this.productID = this.productID.AddLeadingZeros();`

RemoveLeadingZeros

Description	Removes leading zeros from the identifier.
Example	`this.productID = this.productID.RemoveLeadingZeros();`

7.2.7 Basic Data Types

7.2.7.1 Behavior of Data Types

Overview

The scripting language supports basic data types such as string, numeric and Boolean values. In addition, the existing metadata of the underlying SAP cloud platform for core data types (CDTs) and global data types (GDTs) is made available.

The implementation type of the particular data type is taken into account during the mapping of CDTs and GDTs to the basic types of the scripting language. In the studio, you can explore all SAP data types that are released with the public solution model (PSM) and their implementation type in the *Repository Explorer* on the *Data Types* tab. For more information, see Repository Explorer [page 49].

Basic Data Types

The basic data types that are supported by the studio are listed in the following table:

Basic Type	Implementation Type	Properties	XSD Data Type	SAP GDT or SAP CDT, For Example:
Boolean			`xsd:boolean`	`Indicator`
Numeric (FLTP)	INT1	1-byte integer (internal). Value range: 0 to 255. The `MinimumValue` attribute and the `MaximumValue` attribute define the business contract of the data type. These attributes do not limit the value range technically.	`xsd: unsignedByte`	`SMALLNONNEGATIVEINTEGER_DecimalValue`
	INT4	4-byte integer (internal). Value range: -2 (`MinimumValue`), 147,483,648 to +2,147,483,647 (`MaximumValue`) .	`xsd:int`	• `NumberValue` • `IntegerValue`
	DEC	Decimal number with fixed length (`MaximumDigits`) and fixed number of fractions (`MaximumFractions`).	`xsd:decimal`	• `Amount.content` • `Quantity.content` • `Measure.content` • `DecimalValue` • `SMALLNONNEGATIVE_Ratio` • `Percent`
	DF34_DEC	Decimal floating point numbers of this type are represented internally with 34 decimal places in accordance with the IEEE-754-2008 standard (DF34_DEC). Value range: numbers between 1E6145(1E-34 - 1) and -1E-6143 for the negative range, 0 and +1E-6143 and 1E6145(1 - 1E-34) for the positive range. Values lying between these ranges are in the subnormal range and are rounded. Outside the subnormal range, each 34-digit decimal number can be represented precisely by such a decimal floating point number.		

Basic Type	Implementation Type	Properties	XSD Data Type	SAP GDT or SAP CDT, For Example:
Numeric Character	NUMC	Character string containing the digits 0 to 9 only. Valid content of a numeric text field. The `maxLength` property specifies the maximum length of the string.	`xsd:token`	`AccountingPeriodID` `FiscalYearID`
String	CHAR, SSTR, and STRG	Character string containing any alphanumeric characters, including special characters. SAP cloud systems are Unicode systems with the system code page UTF-16. A character has a length of two bytes. The `maxLength` property specifies the maximum length of the string.	• `xsd:token` • `xsd: string` • `xsd:anyURI`	• `Text.content` • `LANGUAGEINDEPENDENT_SHORT_Name` • `LONG_Description.content` • `BusinessTransactionDocumentID` • `ID` • `EmailURI` • `ApprovalStatusCode` • `Amount.currencyCode`
Binary	`RSTR, RAW`	String of bytes	• `xsd:base` • `64 Binary`	`BinaryObject`. In particular: usage of `AttachmentFolder` dependent object
UUID	`XSDUUID_RAW`	Universally unique identifier	`xsd:token`	`UUID`
Date	`DATS`	Gregorian date “YYYY-MM-DD”	`xsd:xsd:Date`	`Date`
GlobalDateTime	`XSDDATETIME_LONG_Z`	Gregorian date and time “2011-12-25T02:30:00Z”	`xsd:DateTime`	`GLOBAL_DateTime`
Time	`TIMS`	Time “HH:MM:SS”	`xsd:time`	`Time`
Duration	`XSDDURATION_ISO`	Duration, for example “P12Y11M2DT4H12M40S”	`xsd:duration`	`Duration`
LanguageCode	`XSDLANGUAGE`	Language code, for example “E”	`xsd:language`	`LanguageCode`

For more information, see the data type definitions of the World Wide Web Consortium (W3C) at http://www.w3.org/TR/xmlschema-2/ .

Validation by the User Interface and Web Service Runtime

The user interface (UI) runtime of the SAP cloud solution validates the user input according to the data type definition. The following data type properties are checked in particular:

- For strings: Maximum length
- For numeric values: Minimum and maximum value
- For Boolean expressions: Usually represented by a checkbox
- For date, time, and duration values: Defined length and content

As a consequence, you do not need to implement validations in your script files to ensure that the user input matches the data type definition. The SAP web service runtime validates the service input according to the data type definition (XSD schema validation).

Internal Variables of the Scripting Language

Internal string variables defined in the scripting language are not restricted in length. “Not restricted” means that the variables are limited by the memory quota of a session, which is defined by the system administrator.

Internal numerical variables defined in the scripting language are of the `DF34_DEC` implementation type.

Example

```
var string = this.ShortText; // string is a unrestricted string

var i = this.IntegerValue; // i is of DF34_DEC type
```

In the following example, a very long text is used:

```
var string = "Lorem ipsum dolor sit amet, consectetur …"
// very long text, containing \"culpa\" at position 1000+ and \"augue\" at
position 2000+
position = string.Find("\"culpa\"");
string.Substring(position); // result: ""culpa" …" (not restricted)
position = string.Find("\"augue\"");
this.ShortNote = string.Substring(position);// result: ""augue" duis …" (40
char)
```

Conversion

Syntax

If variables are typed with data types that belong to the same basic type in the scripting language, they are implicitly converted by the scripting language. For example:

```
this.ID = this.ShortName;

if ( this.Amount.content > this.IntegerValue) …
```

If variables are typed with data types that belong to different basic types in the scripting language, they need to be converted by using conversion functions, for example, `ToString()`, `Numeric.ParseFromString()`, `GlobalDateTime.ConvertToDate()`..

Behavior

If you perform operations on string variables in your script files, you need to ensure that the length of the result strings match the data type definition. For example:

```
// ShortName is typed with LANGUAGEINDEPENDENT_SHORT_Name -> length = 10
// LongName is typed with LANGUAGEINDEPENDENT_LONG_Name -> length = 40
```

```
this.ShortName = this.LongName.Substring(0, 10);
```

> **i** Note
>
> The program terminates when saving if the length of the `LongName` field is greater than 10 characters and no check exists.

If the content of a variable is outside the range that is specified by the data type definition, the following can occur as a consequence:

- Program termination, for example, an overflow error
- A system message
- An implicit conversion, for example, a cutoff

> **i** Note
>
> Variables of unrestricted length, for example, variables of the `LANGUAGEINDEPENDENT_Text` data type, are truncated at a length of 255 characters as soon as they are saved to the database. For texts that are longer than 255 characters, use the `TextCollection` dependent object.

For variables that are typed with numerical data types, rounding is performed according to the definition of the target data type. While the scripting language uses `decfloat34` internally, the values are rounded according to the definition of the data type as soon as the value is assigned to a business object element.

> **Example**
>
> **Note**: The `Quantity.content` data type is specified as a decimal value with 31 digits and 14 fractions.
>
> ```
> this.IntegerValue1 = 1 / 3; // result: 0
> this.IntegerValue2 = 2 / 3; // result: 1
> ```
>
> ```
> var num = 1/3;
> this.IntegerValue3 = num * 3; // result: 1
> this.IntegerValue4 = this.IntegerValue1 * 3; // result: 0
> ```
>
> ```
>  this.Quantity1.content = 1 / 3000000000; // result: 0.00000000033333
> this.Quantity2.content = 2 / 3000000000; // result: 0.00000000066667
> num = 1/3000000000;
> this.Quantity3.content = num * 3000000000; // result: 1
> this.Quantity4.content = this.Quantity1.content * 3000000000; // result:
> 0.99999
> ```

```
this.Quantity5.content = 1 / 3000000000000000000; // result: 0
num = 1/3000000000000000000;
this.Quantity6.content = num * 3000000000000000000; // result: 1
this.Quantity7.content = this.Quantity5.content * 3000000000000000000; //
result: 0
num = this.Quantity5.content;
this.Quantity8.content = num * 3000000000000000000; // result: 0
```

Additional Properties for Identifier Data Types

For identifier data types, additional properties are specified in the GDT definition:

- Upper case conversion
- Alpha conversion
  Alpha conversion fills purely numeric user input like “4711” with leading zeros from the left side to allow better sorting in character fields. For example, if alpha conversion is not used, an alphabetic sorting would lead to sorting results such as: “1”, “10”, “100”, “1000”, “2”, “20”, “200”, and so on.

The UI runtime performs upper case conversion and alpha conversion automatically. The scripting language, however, does not perform these types of conversion automatically. Therefore you need to ensure this in your script files.

Example

`ID` data type (supporting upper case conversion and alpha conversion)

```
var string =
"123456789a123456789b123456789C123456789D123456789E123456789F123456789G1234567
89H123456789G123456789I";
this.ID = string;
position = this.ID.Find("A"); // result: -1
```

The result is `-1` because the content of the `ID` is not automatically converted to upper case. To convert to upper case, use the `ToUpperCase` built-in function, for example:

```
var string =
"123456789a123456789b123456789C123456789D123456789E123456789F123456789G1234567
89H123456789G123456789I";
this.ID = string.ToUpperCase();
position = this.ID.Find("A"); // result: 9
```

Variables Typed with Code Data Types

Code data types can be mapped to the string basic type or the numeric character basic type. The behavior of a variable typed with a code data type, for example, `PriorityCode`, `ApprovalStatusCode` and `Amount.currencyCode`, is as follows:

- The variable can contain alphanumeric characters or numeric characters. The value range is then defined by fixed or configurable code lists.

- A variable can be assigned, compared, and so on, to a string data type (“=”, “==”, “+”) if the variable is typed with a code data type mapped to the string basic type . If the code data type, however, is mapped to the numeric character basic type, you need to use the `ToString` built-in function in order to assign or compare the variable to a string data type.
- The variable cannot be assigned, compared, and so on, to other code data types (“=”, “==”, “+”)

The UI runtime performs a content check of the code list. The scripting language, however, does not perform this check automatically. Therefore you need to ensure this in your script files.

Example

```
var string = "123456789a123456789b123456789C123456789D";
```

```
this.ApprovalStatusCode = string; // result: "12"
// Note: "12" is not an allowed code value.
```

```
// Note: Assignment between IDs is not allowed.
this.ApprovalStatusCode = this.DeliveryPriorityCode;
```

7.3 Front-End Script for the UI Designer

7.3.1 Front-End Script Reference

Overview

The front-end script is the programming language used in the user interface designer (UI designer) to define a set of rules and the logic of the behavior of UI controls. For example, you can write a front-end script to enable an edit button only when a field is selected on the user interface. The front-end script must only be used for user interface related adaptation; it must not contain any business logic.

Front-end scripts are used in the following use cases:

- Dynamic setting of UI properties such as Visible, Enabled, Read-only, Mandatory, and Color
- UI-specific code lists
- Dynamic calls of event handlers (for example, for dynamic OBN links)

Script Execution

All scripts (including calculation rules) are executed in a sequential manner together with all other events by the event queue processor in the UI runtime on the client. Parallel execution of scripts is not possible. For each execution of a script, the scripts environment is recreated. All variables, including global variables, are reset.

> **ℹ Note**
>
> Due to this execution strategy:
>
> - A large number of executing script instances (such as those triggered by the recalculation of properties) or slow scripts can cause the client to react slowly. This is because all scripts in the event queue are processed before event handler operations that are triggered by end user interaction.
> - Events that are triggered by a script — either directly or implicitly due to data manipulation — are also queued and processed only after the script has terminated.

Syntax

The scripts are defined as part of a UI component model. The UI component models are stored and processed as XML files; the script sources are stored as character data sections within these UI component models. The character set of the scripts is based on UTF-8.

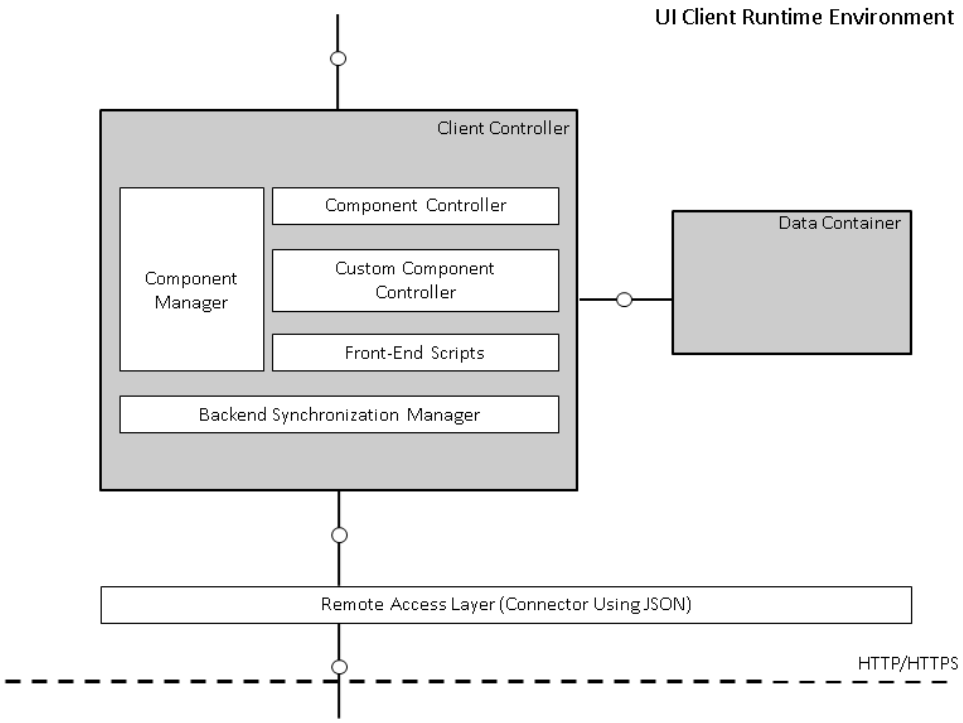
Each script and each block used inside control statements is a sequence of statements. For more information, see Lexical Structure of Front-End Scripts [page 338] and Syntax of Front-End Scripts [page 339].

For more code samples, see Front-End Code Snippets [page 343].

> **⚠ Caution**
>
> Any software coding or code lines/strings ("Code") provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

Interacting with UI Runtime

The following figure provides an overview of the UI client runtime environment that the front-end script interacts with.



Client Controller Overview

Interacting with Runtime Data

The interaction with runtime data is accomplished by accessing members of the global variable `$data`. For more information, see Interacting with Runtime Data [page 349].

Interacting with Runtime Component Controller

Each script is executed as an operation of an event handler and runs within the scope of the UI component controller. The global variable `$controller` provides information about UI component controller's state and methods to interact with the component controller. For more information, see Global Variable: $controller [page 354].

Using the Text Pool

Scripts should never contain any literals that are intended to be presented to the end user, because they cannot be translated. Texts for the end user should also never be created by concatenating variables or

variables and literals, because the order of the elements may have to change in different languages. Instead, you should use modeled texts from the text pool of the UI component.

7.3.2 Lexical Structure of Front-End Scripts

The lexical structure of the front-end scripts contains identifiers, comments, and literals.

Identifiers

Identifiers consist of:

- Letters: a-z, A-Z
- Digits: 0–9
- Special: underscore (_)

Identifiers must start with a letter or an underscore. There is no length restriction for identifiers.

> → Tip
>
> We recommend using reasonable identifiers since parsing times are adversely affected by large identifiers.

Comments

Comments follow Java/C# specification:

- Single line comments start with `//`. The remainder of the line is considered part of the comment.
- Comments delimited by `/*` and `*/`. All content between the delimiters is considered part of the comment.

Literals

- **String Literals**
  String expressions begin and end with double or single quote marks, and these string expressions are subject to backslash escape.
  The scripting engine does not support any escape sequences in the literals.

  > ℹ Note
  >
  > In general, we do not recommend using escape sequences.

- **Numeric Literals**
  - -1234: Numeric integers must only contain decimal digits (other characters such as underscore (_) are not supported).

- 123.56: Decimals must consist of two strings of decimal digits separated by a period (.).

Related Information

Syntax of Front-End Scripts [page 339]

7.3.3 Syntax of Front-End Scripts

The syntax of the front-end script contains variables, statements, and expressions.

Variables

The following variables are supported:

- Global variables prefixed by a `$`-character
  The UI runtime framework automatically adds the following global variables into the script environment:
  - `$controller`: allows interaction with the UI component controller. For more information, see Global Variable: $controller [page 354].
  - `$data`: allows access to and modification of the data of the UI component. For more information, see Interacting with Runtime Data [page 349].
  - `$textpool`: allows the use of modeled texts from the text pool of the UI component.
  - `$eventArgs`: allows accessing event-specific data.
  - `$currentrow`: for calculation rules of controls used inside iterator elements (for example, cell renderers in the table control), `$currentrow` provides access to the list row against which the control is bound.
- Local variables

The basic types that can be used for scripting correspond to the types used by the UI controller:

- `boolean`
- `integer`: a 32–bit integer value
- `decimal`: a 128–bit data type. Compared to floating-point types, the decimal type has a greater precision and a smaller range, which makes it suitable for financial and monetary calculations.
- `string`: string literals
- `date`: date and time
- `data object references`: references to elements from the data container of the UI component.

The initial type of a variable is determined the first time the variable is assigned a value. Afterwards, only values of the same type can be assigned.

Constants are not supported.

Statements

Simple statements are as follows:

assignment statement

Syntax

```
assignment :=
  reference = expression
```

Example

```
tuesday = 'Tuesday'
number = 10
```

print statement

Syntax

```
arguments :=
  expression
  | expression, argument_list

print_statement :=
  print expression
  | print arguments
```

Example

```
print "SAO cloud solution"
print "Value", a
```

Comment: The print() statement, although supported, does not have any practical value for the scripts used in the run-time framework. It is used for internal debugging purposes only.

Control statements are as follows:

if statement

Syntax

```
if_statement:
  if_predicate block end
   | if_predicate block else_block
   | if_predicate block elseif_statement

if_predicate:
  if expression then
  | if expression NEWLINE

else_block:
  else block end

elseif_statement:
  elseif_predicate block end
  | elseif_predicate block else_block
  | elseif_predicate block elseif_statement

elseif_predicate:
  elif expression then
  | elif expression NEWLINE
```

Example

```
if ( true ) then print 'ok' end

if ( true )
  print 'ok'
end

a=1;
if ( a==2 )
   print 'ko' ;
elif ( a==1 )
  print 'ok' ;
else
  print 'ko' ;
end

a=1;
if ( a==2 )
   print 'ko' ;
elif ( a==1 )
  b=3;
  if ( b==2 )
    print 'ko' ;
  elif ( b==1 )
    print 'ko' ;
  else
    print 'ok' ;
  end
else
  print 'else reached' ;
  print 'ko' ;
end
```

unless statement

Syntax

```
unless_statement :=
  unless_predicate block end
  | unless_predicate block else_block

unless_predicate :=
  unless expression   then
  | unless expression NEWLINE;

else_block :=
  else block end
```

Example

```
unless ( false )
  print 'ok'
end
```

ternary statement

Syntax

```
ternary_statement :=
  expression ? statement : statement
```

Example

```
a=1; b=3; c=3;
((a > 0) and (b == 3)) ? (a=1) : (c=2)
```

Comment	In this example, the expression on the left side evaluates to "true". The assignment statement a=1 is executed.

for statement

Syntax	```
	for_statement :=
	  for_in block end
	
	for_in:
	  for reference in expression NEWLINE
	``

Example	``` result = '' for i in 2..9 result = result + i end ```
Comment	Only integer ranges in the form 2..5 (last item included) and 2...5 (last item excluded) are supported. All other expressions will raise a runtime error.

while statement

Syntax	```
	while_statement:
	  while_predicate block end
	
	while_predicate:
	  while expression NEWLINE
	``

Example	```
	i=0
	while i<5
	  print i
	  i=i+1
	end
	``

Expressions

The front-end script interpreter supports the following expressions:

- literals, decimal numbers, integer numbers, constants `true` and `false`
- parentheses to indicate precedence
- method invocations
- variables
- unary expressions
- boolean expressions
- relational expressions
- equality expression
- `not` expression
- range expression: only integer ranges in the form 2..5 (last item included) and 2...5 (last item excluded) are supported.

- #, —, *, /

> **i Note**
>
> All statements are expressions, including methods and variables. The last member pushed onto the stack is considered as the return value of a method, there is no `return` statement. The return value of a calculation rule in the runtime framework is determined in the same way. To return a value from a method or from the front-end script program, use an expression.
>
> **Example**
>
> ```
> a    // will return the value of the variable 'a'
> result = a  // will also return the value of the variable 'a' because the
> assignment statement is an expression and puts 'a' onto the stack
> ```

> **i Note**
>
> The variable `result` has been introduced to facilitate adoption of the runtime framework. It is a common cause of error that after an assignment to `result` (which is a way of moving a value to the stack), another statement moves another value to the stack and this is then used as the result of the calculation rule.

7.3.4 Front-End Code Snippets

Index of Code Samples

The following code snippets are front-end script examples and are grouped thematically:

Lists in Data Container

- Get Lead Selection Data
- Get Lead Selection in a Hierarchical List
- Change Data for All List Entries
- Evaluate a Clicked Link in a List
- Disable a Button Based on a Field in Selected Row in a List
- Set Lead Selection of a List
- Working with Lead Selection in Lists
- Working with Lead Selection in Hierarchical Lists
- Build Up a List and Fill Its Content

Calculation Rules

- Make a Control Read Only Using Calculation Rule
- Accessing Data List Values from Calculation Rules
- Access Description Values of Codes in Scripts and Calculation Rules

Language Constructs

- And/Or/If Clauses
- Break Out of a Loop

Other

- Call Another EventHandler
- Using Textpool for Translatable Text
- Format Amount Field with Right Decimals Depending on Currency Code
- Using $System.Saved
- Prevent the Context from Becoming Dirty
- Check If an Object-Based Navigation (OBN) Target Is Assigned to the User

> ⚠ Caution
>
> Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

Lists in Data Container

Get Lead Selection Data

Description	Gets lead selection data from a list and changes an unbound data model field that is assigned to attribute SelectedVariant of a pane container. Script is executed OnLeadSelectionChange of Data-List:List in Data Model. Result: In an object worklist, the displayed preview pane changes when a new line is selected.

Example	`lead = $data.List.LeadSelectedIndex;`
	`if (lead > -1)`
	`  paymenttype = $data.List.Get(lead).PaymentType;`
	`  if (paymenttype == "18")`
	`  $data.UIState._SelectedVariant = "IncomingCheck"`
	`  else`
	`  $data.UIState._SelectedVariant = "NoData"`
	`  end`
	`end`

Get Lead Selection in a Hierarchical List

Description	Gets lead selection data from a hierarchical list.

Example	`p_lead = $data.List.LeadSelectedIndex;`
	`if (p_lead > -1)`
	`  c_lead = $data.List.Get(p_lead).List.LeadSelectedIndex;`
	`  if (c_lead > -1)`
	`  $controller.HandleEvent("EventOnChild");`
	`  else`
	`  $controller.HandleEvent("EventOnParent");`
	`  end`
	`end`

Change Data for All List Entries

Description	Calculates or changes fields in a list.

Example	`for i in 0..($data.List.Count-1)`
	`  $data.List.Get(i).Column11 = "Hello World " + i`
	`end`

Evaluate a Clicked Link in a List

Description	If a link is clicked in a list, and the lead selection does not change, you need to check the event argument. Note that if you have a simple source side dispatching script (only one variable involved), we recommend using the Eventhandler condition instead of a script.

Example	`row = $eventArgs.RowIdentifier;`
	`if (row)`
	`  paymenttype = $data.DataList.Get(row).PaymentType;`
	`  if (paymenttype == "18")`
	`  $controller.HandleEvent("FireIncomingCheck")`
	`  elseif (paymenttype == "4")`
	`  $controller.HandleEvent("FireOutgoingCheck")`
	`  else`
	`  $controller.HandleEvent("NotSupported")`
	`  end`
	`end`

Disable a Button Based on a Field in Selected Row in a List

Description	You need to create a variable of type Boolean and initial value "true" at root level. Associate the same with enabled property of the button. In the OnClick event of the button, associate an EventHandler of type script and use the following script.

Example	`leadIndex = $data.DataList.LeadSelectedIndex;`
	`if (leadIndex > -1)`
	`  status =`
	`$data.DataList.Get(leadIndex).TaxDeclarationLifeCycleStatusCode`
	`;`
	`  if(status == "2")`
	`  $data.result = false;`
	`  else`
	`  $data.result = true;`
	`  end`
	`else`
	`  $data.result = true;`
	`end`

Set Lead Selection of a List

Description	Note that the selection applies only to the data which is available on the client.
Example	`lead = 2` `$data.DataList.SetLeadSelectedIndex(lead)`

Working with Lead Selection in Lists

Description	Obtains a reference to the selected row in a flat list.
Example	`row = $data.MyList.LeadSelectedRow`

Working with Lead Selection in Hierarchical Lists

Description	Obtains a reference to the selected row in a hierarchical list.
Example	`row = $data.MyList.LeadTreeSelectedRow`

Build Up a List and Fill Its Content

Description	This function may be useful for custom radio button groups. Note that you will have to set the text via a textpool element.

Example	`$data.DataList.Add( )`
	`$data.DataList.Add( )`
	`for i in 0..($data.DataList.Count-1)`
	`  $data.DataList.Get(i).Code = i`
	`  $data.DataList.Get(i).Value = "text" + i`
	`end`

Calculation Rules

Make a Control Read Only using Calculation Rule

Description	In the property dialog of the control, the ReadOnly attribute is available. If you open the editor and chose the Calculation Rule selection, a script window opens. To insert new lines, use the Shift + Enter key combination. In the example below, you must make sure to add IssuerBankCode as an invalidation trigger to execute the script. Note that the value of the last expression is used as the result of a calculation rule. The actual name of the parameter does not matter. This will lead to unwanted side effects if you use any expressions after determining the return value. To avoid this, make sure to do a final expression of the return value at the end of your script. The recommended name for the parameter is result.

Example	`if ($data.IssuerBankCode == "4711")`
	`  result = true;`
	`else`
	`  result = false;`
	`end`
	` // ... any other assignments can happen here`
	`result = result`

Accessing DataList Values from Calculation Rules

Description	If you want to get the current value of a datalist field in a calculation rule, you can use `$controller.Resolve` and the relative path of the field in the datalist. Note that there is limited SAP List Viewer (ALV) functionality for calculated fields in lists. For example, there is no sorting and grouping. In the example below, to ensure that the value is always recalculated correctly, you also have to add the fields LowerBoundaryFiscalYearID_LRL_DE and LowerBoundaryAccountingPeriodID_LRO_DE as invalidation triggers of your calculation rule.

Example	`if ( $controller.Resolve("./TestRunIndicator_LRI_DE") )`
	`  result = $textpool.Lookup("PTK") //Test`
	`else`
	`  result = $textpool.Lookup("PTL") //Update`
	`end`

Access Description Values of Codes in Scripts and Calculation Rules

Description	You can access the code value description by adding `.Description`.

Example	`code = $data.CountryCode`
	`description = $data.CountryCode.Description`
	
	`text = "Code: " + code + ", Description: " + description`
	`$data.CalculatedValue = text`

Language Constructs

And/Or/If Clauses

Description	You can put parts of the and/or/if clause in one row.

Example	`LeadSelection = $data.Interactor1_KA.LeadSelectedIndex`
	`if ( LeadSelection > -1 )`
	`  SelectedRow = $data.Interactor1_KA.Get(LeadSelection)`
	`  if ( SelectedRow.InspectionTypeCode_EP != 5 and SelectedRow.Status_ES == 5  and SelectedRow.KIT_KFQ == "NO" and $controller.Utils.IsInitial(SelectedRow.BRCA_EM) )`
	`    result = false`
	`  else`
	`    result = true`
	`  end`
	`end`

Break Out of a Loop

Description	Useful for conditional breakouts from the loops, such as selecting a checkbox depending on a condition.

Example	`LeadSelection = $data.DataList.LeadSelectedIndex;`
	`KeyFigureLeadSelected =`
	`$data.DataList.Get(LeadSelection).dataField;`
	
	`i = 0;`
	`while i < $data.DataList.RowCount`
	`  if ( $data.DataList.Get(i).dataField ==`
	`KeyFigureLeadSelected )`
	`  break;`
	`  else`
	`  $data.ManageExceptions.Get(LeadSelection).checkbox1 =`
	`true;`
	`  end`
	`  end`
	`i = i + 1;`
	`end`

Other

Call Another Event Handler

Description	The following shows how to call the event handlers "FireIncomingCheck" and "FireAccountingDocument" from a script.

Example	`if (paymenttype == "18")`
	`  $controller.HandleEvent("HandleIncomingCheck")`
	`else`
	`  $controller.HandleEvent("HandleAccountingDocument")`
	`end`

Using Textpool for Translatable Text

Description	Translatable text should not be hard coded in scripts. You should define a new entry in TextPool with parameter binding.
Example	`$data.PreviewTitle = $textpool.Lookup("_tUt43CUQ4E0ZhtAgUA0Em")`

Format Amount Field with Right Decimals Depending on Currency Code

Description	To format an amount according to a currency code, you need a new statement.

Example	`amount = 2.1`
	`currencyCode = "EUR"`
	`result = $controller.Utils.AmountToString(amount, currencyCode)`

Using $System.Saved

Description	You can use $System.Saved in your scripts as follows.

Example	`if ($controller.Utils.IsInitial($data.$System.Saved)) result = true else result = false end`

Prevent the Context from Becoming Dirty

Description	Note that DirtyState is an unbound field that needs to be added under /Root in the data model to save the dirty state temporarily. To prevent the context from becoming dirty, for example on executing a business object action (and thus preventing the work protect popup from being shown), you have to do as follows.

Example	1. Add a script operation before the business object action is executed which saves the current dirty state:
	`$data.DirtyState = $data.$System.IsDirty`
	2. Add another script operation after the business object action is executed which resets the dirty state to the previous value:
	`$data.$System.IsDirty = $data.DirtyState`

Check If an Object-Based Navigation (OBN) Target Is Assigned to the User

Description	You can check if an OBN target configured for a floorplan has a valid UI assigned to the current user.
Example	`$controller.CheckNavigationTarget("ServiceRequestRootCreateWithReference")`

Related Information

Front-End Script Reference [page 335]

7.3.5 Interacting with Runtime Data

Overview

The interaction with runtime data is accomplished by accessing members of the global variable `$data`. The global variable `$data` provides access to the controller data container of the UI component by providing a reference to the `Root` structure.

Structure of Runtime Data

The data container of the UI runtime stores the data of the current UI data in the form of named data elements.

Types of data elements are as follows:

- **Data Structures**
  Data structures may contain other data elements (fields, structures, and lists). Each member inside a structure can be uniquely identified by its name.
  The `Root` of the data container is the entry point to all data available for a UI component instance.
- **Data Fields**
  Data fields are used to hold data values. The following base types are supported:
  - string
  - integer
  - float
  - date
  - boolean
  - binary
    Note that it is not possible to work with binary data fields in scripting.
  - time
    Note that it is not possible to work with time data fields in scripting.
  - datetime
    Note that it is not possible to work with datetime data fields in scripting.
- **Data Lists**
  Lists contain zero or more **List Rows**, which are derived from data structures.

The overall composition structure of the data container is determined at design time. It is not possible to add additional members to a data structure at runtime.

Referencing Runtime Data

> **i** Note
>
> When you use the following rules to work with data container elements, you acquire references to data element instances. The data elements are `mutable`.

Accessing Members of Structures

Data structure members can be accessed using the syntax `<data structure reference>.<member name>`. The result is a reference to the member data element or `null` if the member does not exist.

Note that when a structure data element is created by the data container, all members are initialized:

- Data field members are created with their initial values.
- Data structure members are created and their members are initialized.
- List members are created and hold zero rows.

Examples:

```
// access the field 'NameField' underneat the 'Root' structure
```

```
nameFieldRef = $data.NameField

// access a sub structure
subStructRef = $data.SubStruct

// acess a member of the sub structure
subFieldRef1 = subStructRef.SubField

// directly access a member of the sub structure
subFieldRef2 = $data.SubStruct.SubField
```

Working with Lists

Lists hold zero or more rows of uniform data structures.

Once you have acquired a reference to a list, you may use the following methods and properties to work with the list:

If your list is bound to the backend, the rows that are available in the client may only represent a snapshot of the actual rows the list holds. De-referencing via index is generally discouraged in this case because all indices apply to the current snapshot of rows available in the client.

Method/Property	Returns	Description	Example

Add `()`	Reference to new list row instances	Adds a new list row and returns a reference to resulting list row instance.
		This may only be used with unbound lists. For bound lists, explicitly modeled list operations have to be used.
			`myListRef = $data.MyList`
			`myListRowRef = myListRef.Add()`
			`myListRowRef.ProductTitle = 'SAP Business ByDesign'`
			`myListRowRef.Vendor = 'SAP AG'`

Delete `(rowIdentifier(string))`		Deletes a row identified by the row identifier `rowIdentifier`. This may only be used with unbound lists. For bound lists, explicitly modeled list operations have to be used.	`rowIdentifier = '12'` `$data.MyList.Delete(rowIdentifier)`
Delete `(index(integer))`		Deletes the row with the zero-based `index` from the list. This may only be used with unbound lists. For bound lists, explicitly modeled list operations have to be used.	`$data.MyList.Delete(2)`
Clear `()`		Clears the list (removes all row instances). This may only be used with unbound lists. For bound lists, explicitly modeled list operations have to be used.	`$data.MyList.Clear()`

Get (`rowIdentifier(string)`)	Reference to the row instance	Returns the list row with the specified row identifier or `null` if no such row exists.	`rowRef = $data.MyList.Get('12')`
Get(index(string))	Reference to the row instance	Returns the list row at the zero-based `index` of the list.	`rowRef = $data.MyList.Get(2)`
LeadSelectedIndex	integer	Returns the zero-based index of the lead selected row or `-1` if no lead selection exists. Returns `999999999` if the lead selected row instance is not available on the client (for example, due to paging).	`index = $data.MyList.LeadSelectedIndex`
LeadSelectionNotOnPage	boolean	Returns `true` if the lead selected row instance is not available on the client (for example, due to paging).	`isNotOnPage = $data.MyList.LeadSelectionNotOnPage`
LeadSelectedRow	Reference to the row instance	Returns the lead selected row instance if a lead selection exists and the lead selected row instance is available on the client, which may not always be the case (for example, due to paging)	`rowRef = $data.LeadSelectedRow`
LeadTreeSelectedRow	Reference to the row instance	Returns the lead selected row instance for a hierarchical list along the chain of lead selections in the hierarchical list.	`rowRef = $data.LeadTreeSelectedRow`
RowCount	integer	Returns the number of rows that are currently available on the client for this list. The value may be smaller than `EffectiveRowCount` for paged lists.	`count = $data.RowCount`
EffectiveRowCount	integer	Returns the number of rows of the list. This number may exceed the number of rows which are available on the client.	`count = $data.EffectiveRowCount`

Working with data fields

> **i Note**
>
> The scripting environment is agnostic of the CCTS type system used by the UI layer. Do not use scripting to construct or format any representation of a data field value which is intended to be presented to an end user. Use text pool replacements instead.

Once you have acquired a reference to a data field instance, you may work with it like a normal scripting variable. This means you may assign or retrieve the value.

Additional properties exist for data fields:

Property	Returns	Description	Example
Code	Code value (string)	Returns the code value for a code-list field. The value of the field itself contains the key of the code (which may differ from the code value for hierarchical code lists). Returns `null` if the code list item for the code has not yet been read or if the field does not have a code list.	`codeValue = $data.CodeField.Code`
Description	Description (string)	Returns the description a code-list field (translated human-readable text). Returns `null` if the code list item for the code has not yet been read or if the field does not have a code list.	`codeValue = $data.CodeField.Description`
IsEnabled	Enabled value (boolean)	Returns whether or not the field is enabled. For fields bound to node attributes, the enabled state is determined by the BO / ECO. Client-only fields are always enabled.	`enabledValue = $data.IsEnabled`
Locator	string	Returns the binding expression of the field (including row identifiers if the field is contained in list(s)).	`locator = $data.SomeField.Locator`
ModelLocator	string	Returns the binding expression of the field (without row identifiers if the field is contained in list(s)).	`locator = $data.SomeField.ModelLocator`

7.3.6 Global Variable

7.3.6.1 Global Variable: $controller

Overview

The `$controller` global variable enables script developers to access state information of the UI component controller and to call controller methods.

Controller State Properties

Property	Type	Description	Example

ErrorOccurred	boolean	Returns a boolean value that indicates whether an error has occurred.	`if($controller.ErrorOccurred)`
			`  // block executed on error`
			`end`

TransientErrorOccurred	boolean	Returns a boolean value that indicates whether or not a transient error has occurred. Transient errors are typically caused by input validation errors and have to be resolved by the end user before normal processing can resume.	`if($controller.TransientErrorOccured)`
			`  // block which executes on transient error`
			`end`

GetCurrentLanguage	string	Returns the current logon language of the running session.	`$data.currLanguage = $controller.GetCurrentLanguage();`
Utils	<object>	Returns an object instance which provides general utilities for working with all data types. For more information, see General Utilities for Working with Data [page 355].	`utils= $controller.Utils`
StringUtils	<object>	Returns an object instance which provides utilities for working with strings. For more information, see Utilities for Working with Strings [page 355].	`utils = $controller.StringUtils`

Controller Methods

Method	Description	Example

HandleEvent(eventName(string))	Adds the event with the name `eventName`to the event processing queue of the controller. The method invocation is **not** synchronous.	`$controller.HandleEvent('MyEvent')`
OpenLink(uri(string))	Opens the target `uri` in a new window	`$controller.OpenLink('http://www.sap.com')`
Resolve(bindingExpression(string))	Resolves the data field that is addressed by the specified binding expression. For more information, see Binding Expression [page 358].	`$controller.Resolve('/Root/LastName')`
CheckNavigationTarget(outPortName(string))	Returns a boolean value that indicates whether or not a navigation is possible via the out-port with the name `outPortName`. This value is `true` for all dynamic navigations.	`isactive = $controller.CheckNavigationTarget('MyOutPort')`

7.3.6.2 General Utilities for Working with Data

`Utils` returns an object instance which provides general utilities for working with all data types.

Method	Type	Description	Example

IsInitial(data(object))	boolean	This expression is true if the argument data contains the initial value for its type.	`a=0`
			`if($controller.Utils.IsInitial(a))`
			`  // this block will execute`
			`end`

AmountToString(amount(decimal), currencyCode(string))	string	Returns the amount value as a string formatted according to the currency code. This also ensures that the correct number of decimal digits as defined by the currency is used.	`amount = 2.1`
			`currencyCode = "EUR"`
			`result = $controller.Utils.AmountToString(amount, currencyCode)`

7.3.6.3 Utilities for Working with Strings

An object instance with utilities for working with strings is exposed via the global variable `$controller.StringUtils`.

Property	Type	Description	Example

BeginsWith(text(string), pattern(string))	boolean	Returns a boolean value which indicates whether `text` starts with `pattern`.	`if($controller.StringUtils.BeginsWith('SAP cloud solution', 'SAP'))`
			`  // this block will execute`
			`end`

EndsWith(text(string), pattern(string))	boolean	Returns a boolean value which indicates whether `text` ends with `pattern`.	`if($controller.StringUtils.EndsWith('SAP cloud solution', 'SAP'))`
			`  // this block will not execute`
			`end`

Contains(text(string), pattern(string))	boolean	Returns a boolean value which indicates whether `text` contains `pattern`.	`if($controller.StringUtils.Contains('SAP cloud solution', 'cloud'))`
			`  // this block will execute`
			`end`

AsciiToChar(ascii(integer))	string	Returns the ascii character (as a string) which corresponds to the code `ascii`.	`if(s = $controller.StringUtils.Contains(32)`
			`if(s == ' ')`
			`  // this block will execute`
			`end`

IndexOfString(text(string), pattern(string))	integer	Returns the zero-based first position of `pattern` in `text` or -1, if `pattern` does not occur in `text`.	`position = $controller.StringUtils.IndexOfString('SAP cloud solution', 'cloud')`
			`if(position == 4)`
			`  // this block will execute`
			`end`

Length(text(string))	integer	Returns the number of characters in `text`.	`l = $controller.StringUtils.Length('SAP')`
			`if(l == 3)`
			`  // this block will execute`
			`end`

LeftSubString(text(string), length(integer))	string	Returns the first `length` characters of `text`.	`s = $controller.StringUtils.LeftSubString('SAP ods', 3)`
			`if(s == 'SAP')`
			`  // this block will execute`
			`end`

RightSubString(text(string), length(integer))	string	Returns the last `length` characters of `text`.	`s = $controller.StringUtils.RightSubString('SAP ods', 3)`
			`if(s == 'ByD')`
			`  // this block will execute`
			`end`

Trim(text(string))	string	Returns `text` without leading and trailing blanks.	`s = $controller.StringUtils.Trim(' SAP ods ')`
			`if(s == 'SAP ods')`
			`  // this block will execute`
			`end`

LeftTrim(text(string))	string	Returns `text` without leading blanks.	`s = $controller.StringUtils.LeftTrim('  SAP ods ')`
			`if(s == 'SAP ods ')`
			`  // this block will execute`
			`end`

RightTrim(text(string))	string	Returns `text` without trailing blanks.	`s = $controller.StringUtils.RightTrim('  SAP ods ')`
			`if(s == '  SAP ods')`
			`  // this block will execute`
			`end`

Replace(text(string), oldValue(string), newValue(string))	string	Replaces all occurences of `oldValue` in text to `newValue` and returns the result as a new string.	`s = $controller.StringUtils.Replace('SAP ods', 'ods', 'cloud')`
			`if(s == 'SAP cloud')`
			`  // this block will execute`
			`end`

7.3.6.4 Binding Expression

Using the `Resolve` controller method, script developers can use binding expressions to resolve the value of a field.

```
binding_expression :=
  /Root binding_particles

binding_particles :=
  binding_particle
  | binding_particle binding_particle

binding_particle :=
  /{data element name}
  | /{list name}[{row identifier}]
  | /{list name}/@{row index}
```

Particles

All binding expressions must start with the `/Root` particle and must specify at least one additional particle.

Each particle addresses a member of its parent by name.

- **Example 1:** Assume that a data field named *LastName* is defined under the root of the data model, the expression `/Root/LastName` can be used to address the data field.
- **Example 2:** Assume that a data field named *LastName* is defined as a member of the structure *Address*, which is defined under the root of the data model, the expression `/Root/Address/LastName` can be used to address the data field.

List Rows

List rows can be addressed as follows:

- By providing the row identifier
  Example: Assume that a data field named *LastName* is defined as a member of the list *Addresses*, which is defined under the root of the data model, and the row identifier of the row to be addressed is *4711*, the expression `/Root/Addresses[4711]/LastName` can be used to address the field.

- By providing the (zero-based) row index
  Example: Assume that a data field named *LastName* is defined as a member of the list *Addresses*, which is defined under the root of the data model, and the row identifier of the row to be addressed is the fifth row in the list, the expression `/Root/Addresses/@4/LastName` can be used to address the field.

  > **i** Note
  >
  > Using indices is not a reliable way of addressing rows in lists that may be paged. Unless modeled otherwise, **all** bound lists are paged by default.

- Via the lead selection
  Example: Assume that a data field named *LastName* is defined as a member of the list *Addresses*, which is defined under the root of the data model, the expression `/Root/Addresses/LastName` can be used to address the field of the row of the list *Addresses*, which has the lead selection.

8 Developer Desktop

8.1 Overview of the Developer Desktop

The development tools provided by the SAP Cloud Applications Studio (“studio”) are presented in an integrated development environment (IDE) based on Microsoft Visual Studio. The user interface (UI) of the studio is referred to as the developer desktop. From the developer desktop you access specific features and tools of the development environment.

> ⚠ Caution
>
> Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

Choose a Help Topic for the Developer Desktop

- Business Configuration [page 362]
  Learn how to create and change business topics, business options, and business configuration sets for your solution. Also find out how to configure a business configuration view and assign it to the *Business Configuration* work center and how to check your content in the business adaptation catalog.
  In addition, find information on the tools available in the SAP cloud solution to assist you in creating and editing your business configuration content.
- Business Objects [page 398]
  Learn how to create a business object, define its nodes and relevant elements, create script files, activate your business object, and generate the screens for your business object.
- Extensibility
  - Business Object Extensions [page 438]
    Learn how to create a business object extension, define business logic for a business object extension, screens, reports, and the Enterprise Search.
  - Process Extension Scenarios [page 459]
    A process extension scenario links the data from one business context to other related business contexts. Learn how to create a process extension scenario from a list of predefined extension scenarios.
  - Enhancement Implementations [page 464]
    SAP provides enhancement options based on SAP technology that allow you to enhance the standard behavior of specific applications of the SAP cloud solution without modifying the standard solution. Learn how to create enhancement implementations of SAP enhancement options.
- Reuse Libraries [page 468]
  Learn how to create and define your own reuse libraries and reuse functions to organize complex code in your solution or to reuse your code in script files.
- Code List Data Type [page 472]
  Learn how to define a code list data type for your solution and use it when you create your custom business object or business object extension.
- Notification Rules
  Learn how to enable your custom business object for notification rules.
- Analytics [page 511]
  Learn how to create and change data sources, and create and manage reports, and get informed about analytical report content, transformation rules, and aggregation.
  In addition, find information on the tools available in the SAP cloud solution to assist you in defining your reports.
- Mashups [page 551]
  Learn about creating mashups using the studio and the UI designer.
- Service Integration [page 575]
  Learn how to configure internal communication to communicate with business objects in different deployment units, and create XML file input to import external XML data into a business object.
  In addition, find information on the tool for file input in the SAP cloud solution.
- Web Services [page 586]
  Learn how to create, modify, authorize and test a Web service, and how to remove authorization to a Web service.
- Mass Data Run [page 538]
  Learn how to create a mass data run for your custom business object.
- Print Forms [page 508]
  Learn how to create a print form, add a preview button to it, and test the print form. In addition, find information on the tools available in the SAP cloud solution to assist you in creating and editing your print forms.
- Approval Process [page 493]
  Learn how to create a default approval process, assign it to the UI, and test it.
  In addition, find information on the tools available in the SAP cloud solution to assist you in creating and editing your approval process.

Find out about Debugging, Tracing, Dump Analysis, and Translation

- Debugging [page 629]
  Learn how to use the debugger to observe the runtime behavior of your solution and to locate logic errors. The debugger works with your script files and allows you to break the execution of your solution at specified breakpoints to examine your code and to display values of variables defined in your solution.
- Tracing [page 639]
  Learn how to use tracing to record information about your code as it is being executed. From the *Trace Explorer*, you can run a local trace, run a trace for a business user in the SAP cloud solution, use incident traces for troubleshooting purposes, and view the results of a recorded trace.
- Dump Analysis [page 649]
  The *Dump Analysis* tool window presents dump information in a reader friendly format. Learn how the tool window supports you in finding the root cause of a dump.
- Report an Incident [page 628]
  If you encounter a problem during development, you can report an incident in the SAP cloud solution to request help in solving the problem.
- Translation [page 651]
  Find out how you can translate your solution content into the languages supported by the SAP cloud solution.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

8.2 Business Configuration

8.2.1 Business Configuration Quick Guide

In SAP cloud solutions, business configuration enables prospective or existing customers to evaluate and set up their system in the production environment to meet company-specific requirements. It also allows customers to adapt and optimize their solutions at any time as business needs change.

You must create business configuration content for all solution capabilities that you create using the studio. You create business configuration content in the studio and anchor your solution in the business adaptation catalog (BAC) so that it is available for activation by customers who buy your solution.

> **i Note**
>
> - You **cannot** create business configuration content in solution templates.
> - You can create business configuration content on **your development tenant** only.

Availability

> **⚠ Caution**
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Business and Technical Background

Business Adaptation Catalog (BAC)

The full set of the SAP cloud solution's capabilities is outlined in a central business adaptation catalog (BAC). This catalog organizes and structures the capabilities into a hierarchy of business areas, packages, topics, and options. Solutions created in the studio require business configuration content that then appears as elements (BAC elements) in the catalog and allows administrators to implement solutions in the production environment.

For more information, see here [page 370].

Business Configuration Content

You can define the following business configuration content in the studio:

Business Configuration Content

Type	Use	How To...
Business option (mandatory)	Create a business option when no selective adaptation of business configuration content is required.	...Create a Business Option [page 375]
Business topic and business options	Create a business topic when you need to provide multiple business options to key users.	...Create a Business Topic and Business Options [page 377]
Country	Define a country if your solution requires country-specific scoping for a country that is not supported by the SAP standard cloud solution.	...Create a Country and Business Options [page 380]

Type	Use	How To...
Business configuration set (BC set)	Create a BC set to configure the behavior of business processes in your solution and define code list data types that you can use to model fields in a business object. Create a BC set using an SAP business configuration object (BCO) to configure the behavior of standard business processes in the SAP cloud solution.	...Create a BC Set Using an SAP BCO [page 383] ...Create a BC Set Using a Custom BCO [page 387]
Business configuration object (BCO)	Create a custom BCO to define solution-specific business configuration content on which you can base a BC set.	...Create a Business Configuration Object [page 386]
Business configuration view (BC view)	Create a BC view to allow key users to change the values defined in a BC set during fine-tuning.	...Create a Business Configuration View [page 389]
Fine-tuning project template	Create a fine-tuning project template to provide sample content for a specific scenario that is defined by SAP.	...Create a Fine-Tuning Project Template [page 392]

Defining Business Configuration Content

When you create a solution in the studio, you must create business configuration content that contains at least one business option to allow customers who buy your solution to activate the solution in the production environment.

For more information, see here [page 371].

Configurable Code Lists

You can use the business configuration wizards to create configurable code lists.

For more information, see .

Maintenance of Business Configuration Content

A customer-specific solution is in maintenance mode, if it has been assembled and downloaded; the solution status then is *Assembled*. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.

For information about creating a patch, see Patches [page 178].

For information about the change and delete restrictions for business configuration content in a solution that is in maintenance mode, see Maintenance of Business Configuration Content [page 372].

Customer Tools for Business Configuration

If you want to know more about how customers configure their solutions and work with business configuration content, take a look at the "Configuring Your SAP Solution" topic in the documentation in your SAP cloud solution.

Tasks

Tasks in the Business Configuration Wizard

You perform the following tasks using the *Business Configuration Wizard*:

- **Create a Business Option**

  You can create a business topic to group business options and provide multiple business options for customers to customize your solution.

  For more information, see here [page 375].
- **Create a Business Topic and Business Options**

  You can create a business topic to group business options and provide multiple business options for customers to customize your solution.

  For more information, see here [page 377].
- **Create a Country and Business Options**

  You can define a country and create one or more business options, if your solution requires country-specific scoping for a country that is not supported by the SAP cloud solution.

  For more information, see here [page 380].
- **Change a Business Option or a Business Topic**
  1. In the studio, open the solution that contains the business option or business topic you want to change.
  2. In the *Solution Explorer*, right-click the solution and select *Open*.
     The *Business Configuration Wizard* opens.
  3. Make your changes and then click *Save*.
     To change a business option that you added to a business topic, in the *Create Business Options and Assign Solution Content* step, use the *Change Business Option* function.
- **Delete a Business Option or a Business Topic**
  1. In the studio, open the solution that contains the business option or business topic you want to change.
  2. In the *Solution Explorer*, right-click the solution and do one of the following:
     - If you have created only one business option, select *Delete*.
     - To delete a business topic and all business options you added to the business topic, select *Delete*.

- To delete a business option that you added to a business topic without deleting the business topic:
  1. Select *Open*.
     The *Business Configuration Wizard* opens.
  2. In the *Create Business Options and Assign Solution Content* step, select the business option you want to delete and then click *Delete Business Option*.

> **i Note**
>
> For information about the change and delete restrictions for business configuration content in a solution that is in maintenance mode, see Maintenance of Business Configuration Content [page 372].

Create Business Configuration Set

Solution Explorer > *Business Configuration* > *Create Business Configuration Set*

You perform the following tasks using the *Business Configuration Set Wizard*:

- **Create a BC Set Using an SAP BCO**

  You can create a business configuration set (BC set) using an SAP business configuration object (BCO) to configure the behavior of standard business processes in the SAP cloud solution. If you create an implementation of an enhancement option which is for single use only, you can also define a BC set using an SAP BCO so that you can enhance the filter values.

  For more information, see here [page 383].
- **Create a BC Set Using a Custom BCO**

  You can create a business configuration set (BC set) using a business configuration object (BCO) that you have defined in your solution.

  For more information, see here [page 387].
- **Change a Business Configuration Set**
  1. In the studio, open the solution that contains the business configuration set (BC set) you want to change.
  2. If the BC set is based on a business configuration object (BCO) defined in the solution and you want to add or change a field, change the referenced BCO by doing the following:
     1. In the *Solution Explorer*, clean all BC sets that are based on the BCO by right-clicking the `.bcc` files and selecting *Clean*.
     2. Double-click the `.bco` file.
        The BCO opens in a document window.
     3. On the *Fields* tab, add or change one or more fields and then save your changes.
     4. In the *Solution Explorer*, right-click the `.bco` file and select *Activate*.
        You can only activate a BC set that is based on a BCO that you changed by opening the BC set in a document window and saving it.
  3. In the *Solution Explorer*, double-click the `.bcc` file.
     The *Business Configuration Set Wizard* opens.
  4. Make your changes, for example, by adding or changing the field values of any fields you add or changed in the BCO.
     You cannot change the name of a BC set.

5. In the *Solution Explorer*, right-click the `.bcc` file and select *Activate*.

> **i Note**
>
> For information about the change and delete restrictions for business configuration content in a solution that is in maintenance mode, see Maintenance of Business Configuration Content [page 372].

Create Business Configuration Object

You perform the following tasks using the *Business Configuration Object Wizard*:

- **Create a Business Configuration Object**
  In the *Solution Explorer*, click the *Add New Item* button and select `Business Configuration Object`.

  You can create a business configuration object (BCO) and use it as the basis of a business configuration set (BC set).

  For more information, see here [page 386].
- **Change a Business Configuration Object**
  1. In the *Solution Explorer*, clean all BC sets that are based on this BCO by right-clicking the `.bcc` files and selecting *Clean*.
  2. In the *Solution Explorer*, double-click the `.bco` file.
     The business configuration object opens in a document window.
  3. On the *Fields* tab, add or change one or more fields and then save your changes.
  4. In the *Solution Explorer*, right-click the `.bco` file and select *Activate*.
- **Delete a Business Configuration Object**
  In the *Solution Explorer*, right-click the `.bco` file and select *Delete*.

> **i Note**
>
> To delete a business configuration object that is used in a business configuration set, you must first delete the BC set.

> **i Note**
>
> For information about the change and delete restrictions for business configuration content in a solution that is in maintenance mode, see Maintenance of Business Configuration Content [page 372].

Create Business Configuration View

You perform the following task in the studio and in the UI designer:

- **Create a Business Configuration View**
  In the *Solution Explorer*, click the *Add New Item* button and select *Business Configuration View*.
  For more information, see here [page 389].

Create a Tax Decision Tree (available for SAP Business ByDesign only)

You can create a tax decision tree to automatically determine a tax event for a country that does not have an SAP country version in SAP Business ByDesign.

For more information, see Create a Tax Decision Tree [page 394].

Create a Fine-Tuning Project Template (available for SAP Business ByDesign only)

You perform the following tasks in the studio and in the SAP cloud solution in the *Business Configuration* work center:

- **Create a Fine-Tuning Project Template**
  In the *Solution Explorer*, click the *Add New Item* button and select *Implementation Project Template*.

  You can create a fine-tuning project template in the studio as part of a solution. Fine-tuning project templates provide sample content for a specific scenario that is defined by SAP, such as Financials Chart of Account. The fine-tuning activities you need to complete for the template are explicitly listed.

  For more information, see Create a Fine-Tuning Template [page 392].
- **Complete an Implementation Project Template**

  Solution Explorer > **.bct* > *Complete*

  1. In the studio, open the solution that contains the implementation project template you want to complete.

     > **i Note**
     >
     > You can complete a project once the design has been accepted.

  2. In the *Solution Explorer*, right-click the `.bct` file of the active implementation project template and select *Complete*.
     Under *Edit Activities*, the activities are no longer displayed as links.
  3. Optional: Check the status of the implementation project template in the studio by doing the following:
     1. Open the implementation project template in the studio by double-clicking the `.bct` file.
     2. Check that the status of the implementation project template is *Completed*, that a completion date is displayed, and that you can no longer change the description.
- **Reopen an Implementation Project Template**

  Solution Explorer > **.bct* > *Reopen*

  1. In the studio, open the solution that contains the implementation project template you want to reopen.
  2. In the *Solution Explorer*, right-click the `.bct` file of the completed implementation project template and select *Reopen*.
  3. Optional: Check the status of the implementation project template in the studio by doing the following:
     1. Open the implementation project template in the studio by double-clicking the `.bct` file.
     2. Check that the status of the implementation project template is *Active* and that no completion date is displayed.

4. Optional: Check the status of the implementation project template in the SAP cloud solution by doing the following:
    1. In the SAP cloud solution, choose *Business Configuration* > *Implementation Projects* and select your project.
    2. Check that the *Edit Project Scope* and *Open Activity List* functions are available and that the status of the project is *Started*.

Check Content in the Business Adaptation Catalog

> **i Note**
>
> To check content in the business adaptation catalog (BAC), for example, for testing purposes, you must be assigned the Business User role and have access rights for the *Business Configuration* work center. For more information, see User Setup Quick Guide [page 33].

1. In the SAP cloud solution, in the *Business Configuration* work center, open the *Implementation Projects* view.
2. Select a project and click *Edit Project Scope*.
   The *Edit Project Scope* guided activity is displayed. The business topic or business option you created using the *Business Configuration Wizard* is displayed in the *Scoping* phase under *Scoping Element*. The description and scoping questions you defined in the *Business Configuration Wizard* are displayed in the *Questions* phase.

Deploy Business Configuration

In the *Solution Explorer*, click the solution and select `Deploy Business Configuration`.

You deploy your business configuration content be able to use it in a solution. The system makes the parameter values available in the development environment for testing.

> **i Note**
>
> If you have defined BAC elements for your solution, a dialog box opens when you use the *Deploy Business Configuration* function. You have to select one of the following options:
>
> - Deploy all business configuration content for your solution
> - Deploy only the business configuration content for your solution that is assigned to the business options you have selected in the business adaptation catalog
>
> This allows you to test both partial and complete activation of your business configuration content in the development environment.

8.2.2 Business and Technical Background

8.2.2.1 Business Adaptation Catalog

Overview

The full set of the SAP cloud solution's capabilities is outlined in a central business adaptation catalog (BAC). This catalog organizes and structures the capabilities into a hierarchy of business areas, packages, topics, and options. Solutions created in the studio require business configuration content that then appears as elements (BAC elements) in the catalog and allows administrators to implement solutions in the production environment.

Customers work with the SAP cloud solution in the production environment after it has gone live. This environment contains all elements required to run the customer's business processes using the SAP cloud solution and customers can adapt the solution in the *Business Configuration* work center.

Features

When you create a solution in the studio, you must create business configuration content that contains at least one business option to allow customers who buy your solution to activate the solution in the production environment.

A business option represents the most detailed decision level involved in defining or adapting a solution capability. Business options can be hidden or can appear in scoping, fine-tuning, and/or the solution proposal. Each business option specifies a particular way to perform a function. Each business option belongs to a business topic. For information about how to create a business option, see Create a Business Option [page 375].

Optionally, you can also create the following business configuration content:

- Business topic
  A business topic relates to specific functions within an application area. In the standard SAP solution, examples are Account Management and Time Administration. Each business topic belongs to a business package and contains a number of business options.
  For information about how to create a business topic, see Create a Business Topic and Business Options [page 377].
- Business configuration set (BC set)
  A BC set is a technical container for the configuration settings and corresponding values that are associated with a business option. The content contained within BC sets is deployed to the runtime environment where it is loaded into the underlying configuration tables. Each configuration setting within a BC set has attributes which specify whether it is hidden, visible, or changeable for the customer.
  For information about how to create a BC set using an SAP business configuration object (BCO), see Create a BC Set Using an SAP BCO [page 383].
- Business configuration view (BC view)
  To allow administrators to change the values of a BC set during fine-tuning, you can create a BC view and assign this view to the *Business Configuration* work center.
  For information about how to create a BC view, see Create a Business Configuration View [page 389].

Related Information

Business Configuration Quick Guide [page 362]

8.2.2.2 Defining Business Configuration Content

Overview

When you create a solution in the studio, you must create business configuration content that contains at least one business option to allow customers who buy your solution to activate the solution in the production environment.

You can assign your business option to a standard business topic or you can create a business topic to which you add your business options.

Prerequisites

- You have created and opened a solution in the studio. For more information, see Create a Customer-Specific Solution [page 177].
- You understand the dependencies that exist between the elements in the business adaptation catalog and the constraints that are applied by the selections that administrators make during scoping.
- To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the *Business Configuration* work center. For more information, see User Setup Quick Guide [page 33].

Process Flow

1. In the studio, you create one or more business options.
   If you create more than one business option, you group your business options in a business topic. For information about how to create one business option, see Create a Business Option [page 375]. For information about how to create more than one business option, see Create a Business Topic and Business Options [page 377].
2. You define your business options.
   1. You provide a scoping question and details about each business option.
      During scoping, administrators select or deselect a business option by answering its associated scoping question. Scoping questions are located in the business adaptation catalog (BAC) at the business topic level. A scoping question is displayed for each business option in the BAC. In the SAP cloud solution, customers can review the details you provide about BAC elements on the *Overview* and *Relevance* tabs.
   2. If you want to assign a business configuration set (BC set) to the business option, you specify whether you want customers to be able to adapt the field values of the BC set during fine-tuning.

When an administrator of the SAP solution selects a business option in the scoping phase that is relevant for fine-tuning, they can adjust the values of the business configuration sets (BC sets) assigned to this business option by performing a fine-tuning activity.

3. You select additional functionality required by your solution by selecting the relevant business topics from the business adaptation catalog list.
4. You define where you want the business configuration content to be available in the business adaptation catalog.
   For more information about the business adaptation catalog, see Business Adaptation Catalog [page 370] and "Configuring Your SAP Solution" in the documentation in your SAP cloud solution.

3. Optional: You create a BC set and assign it to a business option.
   You specify whether administrators are able to add or delete the values of specific fields in fine-tuning.
   For more information, see
   - Create a BC Set Using an SAP BCO [page 383]
   - Create a BC Set Using a Custom BCO [page 387]
4. Optional: You create a business configuration view (BC view) for a BC set using the UI designer and assign this view to the *Business Configuration* work center.
   For more information, see Create a Business Configuration View [page 389].

   > **i Note**
   >
   > You can only create a BC view for a BC set that uses a custom BCO. You cannot create a BC view for an SAP BCO.

5. You activate your business configuration content.
   The system generates the required business configuration object(s) and data type(s).
6. You trigger deployment of your business configuration content to be able to use the content in your solution.

   > **i Note**
   >
   > When you use the *Deploy Business Configuration* function, a dialog box opens. You have to select one of the following options:
   >
   > - Deploy all business configuration content for your solution
   > - Deploy only the business configuration content for your solution that is assigned to the business options you have selected in the business adaptation catalog
   >
   > This allows you to test both partial and complete activation of your business configuration content in the development environment.

7. You check that your business configuration content appears in the BAC in the SAP cloud solution.
   For more information, see Business Configuration Quick Guide [page 362].

8.2.2.3 Maintenance of Business Configuration Content

Overview

A customer-specific solution is in maintenance mode, if it has been assembled and downloaded; the solution status then is *Assembled*. You can make changes to the solution in a patch; however, you can only make

restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.

Change and Delete Restrictions

The following tables give an overview of the change and delete restrictions for business configuration content in a solution that is in maintenance mode.

Maintenance of Business Topics

Action	Allowed
Change *Type of Business Configuration*	No
Change *Name*	No
Change *Description*	Yes
Change *Anchor*	No
Change *Required Scope*	With restrictions: You can remove the required scope, but you cannot add required scope.
Change assigned business options	With restrictions: see "Maintenance of Business Options" below
Delete assigned business options	No
Change content assigned to business options	No
Create (and delete) a new business option	Yes

Maintenance of Business Options

Action	Allowed
Change *Name*	No
Change *Description*	Yes
Change *Visible in Scoping* setting	With restrictions: • If the *Visible in Scoping* option is not selected, selection is allowed. • If the *Visible in Scoping* option is selected, removal is not allowed.
Change *Visible in Fine-Tuning* setting	Yes
Change *Mandatory* setting	No

Action	Allowed
Change *Scoping Question / Statement*	Yes
Change *Anchor*	No

Maintenance of Business Configuration Objects

Action	Allowed
Delete business configuration object	Yes
Change *Name*	No
Change *Description*	Yes
Change *Create Code List Data Type* setting	No
Change or delete fields	No
Add or delete new fields	Yes
Create (and delete) a new business configuration object	Yes

Maintenance of Business Configuration Sets

Action	Allowed
Delete business configuration set	Yes
Change *Business Configuration Object Type*	No
Change *Name*	No
Change *Description*	No
Change *Business Configuration Object*	No
Business configuration set field values: Change or delete rows	No
Business configuration set field values: Add or delete new rows	Yes
Create (and delete) a new business configuration set	Yes

Maintenance of Business Configuration View

Action	Allowed
Delete business configuration view	Yes

Related Information

Business Configuration Quick Guide [page 362]
Lifecycle Management of Customer-Specific Solutions [page 143]
Maintenance Mode [page 185]

8.2.3 Tasks

8.2.3.1 Create a Business Option

Overview

You can create a business topic to group business options and provide multiple business options for customers to customize your solution.

If you want to create more than one business option, you need to create a business topic first to group the business options. For more information, see Create a Business Topic and Business Options [page 377].

Prerequisites

- You have created and opened a solution in the studio. For more information, see Create a Customer-Specific Solution [page 177].
- To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the *Business Configuration* work center. For more information, see User Setup Quick Guide [page 33].

Procedure

1. In the *Solution Explorer*, do the following:
   1. Click the *Add New Item* button.
      The *Add New Item dialog* opens.
   2. Select *SAP* > *Business Configuration* > *BAC Elements* and click *Add*.
      The *Business Configuration Wizard* opens.
2. In the *Create and Anchor Elements in the Business Adaptation Catalog* step, do the following:
   - Under *Type of Business Configuration*, select *Business Option*.
   - Under *Basic Information*, do the following:
     - Enter a name and a description for the business option and then enter a scoping question.
       Your scoping question indicates the functionality of the business option and allows administrators to decide whether to select the business option during project implementation.

- Optional: Select *Visible in Fine-Tuning* if you want to assign a business configuration set (BC set) to the business option and you want customers to be able to adapt the field values of the BC set during fine-tuning; then under *Fine-Tuning*, do the following:
  1. If customers must complete a fine-tuning activity before the solution goes live, select *Mandatory for Go-Live*.
  2. Enter a description for the fine-tuning activity.

  > **Note**
  >
  > A fine-tuning activity is only mandatory for go-live in the customer system if a BC set that was created using a custom business configuration object (BCO) is assigned to the business option. In this case, you must also create a BC view. For more information about creating a BC view, see Create a Business Configuration View [page 389].

- On the *Overview* tab, write a text that describes the function that this element provides and the benefits of using this element. Aim to write no more than three to five sentences. However, when additional valuable information is available, you should include it.

  > **Example**
  >
  > **Example:** The `Foreign Currency Remeasurement Methods for Cash` business option provides a procedure for converting the value of cash balances held in foreign currency to the company currency at a key date, for example, at the end of the month.

- On the *Relevance* tab, write a text that describes when or why the customer should select this element and what are the implications of selecting or not selecting this element. Explain the relevance of this element in terms of real-world business requirements.

  > **Example**
  >
  > **Example:** The `Foreign Currency Remeasurement Methods for Cash` business option is relevant for companies that have cash accounts stated in foreign currencies.automatically included in your scope.

- Under *Business Adaptation Catalog (BAC)*, do the following:
  - Next to the *Anchor* field, click *Select* and anchor your business topic in the BAC by expanding the tree and selecting a business topic or business option group.
  - Optional: Next to the *Required Scope* field, click *Select* and then select the standard BAC elements in the cloud solution that you require for your solution by expanding the tree and selecting one or more business packages or business topics.
  - Click *Next*.

3. In the *Review* step, check that the information you have entered is correct and then click *Finish*.
   The system assigns all content that is relevant for business configuration such as business objects and business configuration sets (BC sets) to your business option. The content is displayed under the business option.
4. In the *Solution Explorer*, right-click the `.bac` file and then select *Activate* to make the business option available in the BAC.

Result

The system has added the business option to the business adaptation catalog (BAC). You can check the result by logging on to the cloud solution as a business user. You select an implementation project in the *Business Configuration* work center and edit the project scope. The *Scoping* step displays the BAC elements and you can navigate to the business option you created.

Related Information

Business Configuration Quick Guide [page 362]

8.2.3.2 Create a Business Topic and Business Options

Overview

You can create a business topic to group business options and provide multiple business options for customers to customize your solution.

Prerequisites

- You have created and opened a solution in the studio. For more information, see Create a Customer-Specific Solution [page 177].
- To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the *Business Configuration* work center. For more information, see User Setup Quick Guide [page 33].

Procedure

1. In the *Solution Explorer*, do the following:
   1. Click the *Add New Item* button.
      The *Add New Item dialog* opens.
   2. Select *SAP* > *Business Configuration* > *BAC Elements* and click *Add*.
      The *Business Configuration Wizard* opens.
2. In the *Create and Anchor Elements in the Business Adaptation Catalog* step, do the following:
   - Under *Type of Business Configuration*, select *Business Topic and Business Options*.

> **Note**
>
> If you need to create only one business option, under *Type of Business Configuration*, select *Business Option*. For information about how to create one business option for a solution, see Create a Business Option [page 375].

- Under *Basic Information*, do the following:
  - Enter a name and a description for the business topic.
  - On the *Overview* tab, write a text that describes the function that this element provides and the benefits of using this element. Aim to write no more than three to five sentences. However, when additional valuable information is available, you should include it.

    > **Example**
    >
    > **Example:** The `Bank Directory File Upload` business topic allows you to upload a bank directory from a provider of your choice into SAP's cloud solution. Before uploading the file, you must ensure that the contents of the file, namely the bank data, matches the order prescribed in the SAP format.

  - On the *Relevance* tab, write a text that describes when or why the customer should select this element and what are the implications of selecting or not selecting this element. Explain the relevance of this element in terms of real-world business requirements.

    > **Example**
    >
    > **Example:** The `Bank Directory File Upload` business topic is relevant if you wish to upload a bank directory from a provider of your choice.

- Under *Business Adaptation Catalog (BAC)*, do the following:
  - Next to the *Anchor* field, click *Select* and anchor your business topic in the BAC by expanding the tree and selecting a business package.
  - Optional: Next to the *Required Scope* field, click *Select* and select the standard BAC elements in the cloud solution that you require for your solution by expanding the tree and selecting one or more business packages or business topics. If you reference SAP content in BC sets, you need to select the SAP business topic to which the corresponding business option is assigned as the required scope. The system checks this selection before test deployment.

3. In the *Create Business Options and Assign Solution Content* step, create your business options by doing the following:
   1. Click *Create Business Option*.
      The *Create Business Option* dialog box opens.
   2. In the *Create Business Option* dialog box, under *Basic Information*, do the following:
      1. Enter a name and a description for the business option.
      2. On the *Overview* tab, write a text that describes the function that this element provides and the benefits of using this element. Aim to write no more than three to five sentences. However, when additional valuable information is available, you should include it.

         > **Example**
         >
         > **Example:** The `Foreign Currency Remeasurement Methods for Cash` business option provides a procedure for converting the value of cash balances held in foreign currency to the company currency at a key date, for example, at the end of the month.

3. On the *Relevance* tab, write a text that describes when or why the customer should select this element and what are the implications of selecting or not selecting this element. Explain the relevance of this element in terms of real-world business requirements.

> **Example**
>
> **Example:** The `Foreign Currency Remeasurement Methods for Cash` business option is relevant for companies that have cash accounts stated in foreign currencies.automatically included in your scope.

3. Optional: If you want the business option to appear in the BAC, select *Visible in Scoping* and under *Scoping*, do the following:
   - If your business option provides functionality that is essential for your solution, select *Mandatory* and enter a scoping statement that describes the functionality of the business option. By default, the business option will be in scope when a customer selects the business topic in the BAC.
   - If your business option is not mandatory, enter a scoping question. Your scoping question indicates the functionality of the business option and allows administrators to decide whether to select the business option during project implementation.
4. Optional: Select *Visible in Fine-Tuning* if you want to assign a business configuration set (BC set) to the business option and you want customers to be able to adapt the field values of the BC set during fine-tuning; then under *Fine-Tuning*, do the following:
   1. If customers must complete a fine-tuning activity before the solution goes live, select *Mandatory for Go-Live*.
   2. Enter a description for the fine-tuning activity.

> **Note**
>
> A fine-tuning activity is only mandatory for go-live in the customer system if a BC set that was created using a custom business configuration object (BCO) is assigned to the business option. In this case, you must also create a BC view. For more information about creating a BC view, see Create a Business Configuration View [page 389].

5. Under *Anchor Business Option below*, you can either accept the selection of *Header Element of the Solution* or select *SAP Element* and then *Select Anchor* to anchor your business option in the BAC. Expand the tree and select your business topic, an SAP business topic or business option group and click *OK*.
6. Repeat the steps above for each business option and click *Next*.
   The *Create Business Options and Assign Solution Content* step now displays all business options that you created on the right.

4. In the *Create Business Options and Assign Solution Content* step, assign solution content to each business option by using the arrow buttons or by means of drag-and-drop.
   The system displays the solution content you assign under the business option. You can also use the arrow buttons or drag-and-drop to unassign solution content.

> **Caution**
>
> You must assign all solution content such as business objects and BC sets to a business option before you trigger test deployment of your solution. If dependencies exist between BC sets defined in your solution. for example, a value defined in one BC set is referenced in a second BC set, you must assign both BC sets to the same business option or you must assign the referenced BC set to a mandatory business option.

5. In the *Review* step, check that the information you have entered is correct and then click *Finish*.
6. In the *Solution Explorer*, right-click the `.bac` file and then select *Activate* to make the business topic available in the BAC.

Result

The system has added the business topic to the business adaptation catalog (BAC). You can check the result by logging on to the cloud solution as a business user. You select an implementation project in the *Business Configuration* work center and edit the project scope. The *Scoping* step displays the BAC elements and you can select the business topic that you created.

If you have created all business configuration content your solution requires, you can trigger deployment of your business configuration content to be able to use the content in your solution. To do this, right-click your solution and select *Deploy Business Configuration*.

> **i** Note
>
> When you use the *Deploy Business Configuration* function, a dialog box opens. You have to select one of the following options:
>
> - Deploy all business configuration content for your solution
> - Deploy only the business configuration content for your solution that is assigned to the business options you have selected in the business adaptation catalog
>
> This allows you to test both partial and complete activation of your business configuration content in the development environment.

Related Information

Business Configuration Quick Guide [page 362]

8.2.3.3 Create a Country and Business Options

Overview

You can define a country and create one or more business options, if your solution requires country-specific scoping for a country that is not supported by the SAP cloud solution.

Prerequisites

- You have created and opened a solution in the studio. For more information, see Create a Customer-Specific Solution [page 177].
- To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the *Business Configuration* work center. For more information, see User Setup Quick Guide [page 33].

Procedure

1. In the *Solution Explorer*, do the following:
    1. Click the *Add New Item* button and click *Add*.
    The *Add New Item dialog* opens.
    2. Select *SAP* > *Business Configuration* > *BAC Elements*.
    The *Business Configuration Wizard* opens.
2. In the *Create and Anchor Elements in the Business Adaptation Catalog* step, do the following:
    1. Under *Type of Business Configuration*, select *Country and Business Options*.
    2. Under *Basic Information*, enter a name and description and then select the country for which your solution is relevant.
3. In the *Create Business Options and Assign Solution Content* step, create one or more business options for your country by doing the following:
    1. Click *Create Business Option*.
    The *Create Business Option* dialog box opens.
    2. In the *Create Business Option* dialog box, under *Basic Information*, do the following:
        - Enter a name and a description for the business option.
        - On the *Overview* tab, write a text that describes the function that this element provides and the benefits of using this element. Aim to write no more than three to five sentences. However, when additional valuable information is available, you should include it.

            > **Example**
            >
            > The Time Recording Integrated with Procurement business option allows you to use time recording for third party service agents, such as consultants and service technicians. The working times of service agents can then be allocated to procured services and purchase orders. The recorded times are taken into account in invoice verification.

        - On the *Relevance* tab, write a text that describes when or why the customer should select this element and what are the implications of selecting or not selecting this element. Explain the relevance of this element in terms of real-world business requirements.

            > **Example**
            >
            > The Time Recording Integrated with Procurement business option is relevant if you want to consider the working time recorded by service agents for invoice verification. If the Services with Time Sheet Recordings for Projects business option (within the Service and Non-Stock Material Procurement business topic) is selected, this business option is automatically included in your scope.

3. Optional: If you want the business option to appear in the BAC, select *Visible in Scoping* and under *Scoping*, do the following:
    1. If your business option provides functionality that is essential for your solution, select *Mandatory* and enter a scoping statement that describes the functionality of the business option.
    If your business option is not mandatory, enter a scoping question. Your scoping question indicates the functionality of the business option and allows administrators to decide whether to select the business option during project implementation.

    > ⚠ Caution
    >
    > When you define a country, you must create at least one business option that is visible in scoping.

    2. Click *Select Anchor* and anchor your business option in the BAC by expanding the tree and selecting a business topic or a business option group.
4. Optional: Select *Visible in Fine-Tuning* if you want to assign a business configuration set (BC set) to the business option and you want customers to be able to adapt the field values of the BC set during fine-tuning; then under *Fine-Tuning*, do the following:
    1. If customers must complete a fine-tuning activity before the solution goes live, select *Mandatory for Go-Live*.
    2. Enter a description for the fine-tuning activity.

    > i Note
    >
    > When you define a business configuration set (BC set), you must also specify whether administrators are able to add or delete the values of specific fields. For more information about creating a BC set, see Create a BC Set with a Custom BCO [page 387].

5. Repeat the steps above for each business option.
    The *Create Business Options and Assign Solution Content* step now displays all business options that you created for the country on the right.

4. In the *Create Business Options and Assign Solution Content* step, assign solution content to each business option by using the arrow buttons or by means of drag-and-drop.
    The system displays the solution content you assign under the business option. You can also use the arrow buttons to unassign solution content. Click *Next*.

    > ⚠ Caution
    >
    > You must assign all solution content such as business objects and BC sets to a business option before you trigger test deployment of your solution. If dependencies exist between BC sets defined in your solution. for example, a value defined in one BC set is referenced in a second BC set, you must assign both BC sets to the same business option or you must assign the referenced BC set to a mandatory business option.

5. In the *Review* step, check that the information you have entered is correct and then click *Finish*.
6. In the *Solution Explorer*, right-click the `.bac` file and then select *Activate* to make the country and its business options available in the BAC.
7. Check that the system has added the country and the business options you created to the business adaptation catalog (BAC) by doing the following:
    1. Log on to the cloud solution as a business user.
    2. In the *Business Configuration* work center, open the *Implementation Projects* view.

3. Select a project and click *Edit Project Scope*.
   The *Edit Project Scope* guided activity is displayed.
4. In the *Country and Type of Business* step, click *Edit Countries* and check that the country you created is display in the list of available countries.
5. In the *Implementation Focus* step, select *Complete Solution*.
6. In the *Scoping* step, check that the visible business options you created using the *Business Configuration Wizard* are displayed.
   The business options are displayed in the *Scoping Element* column under the business topic you selected as an anchor. The description and scoping questions you defined in the *Business Configuration Wizard* are displayed in the *Questions* step.

Result

The system has added your country and the business options you created to the business adaptation catalog (BAC) and you have checked the result in the cloud solution. The country description is displayed in the business adaptation catalog in the *Country and Type of Business* step.

If you have created all business configuration content your solution requires, you can trigger deployment of your business configuration content to be able to use the content in your solution. To do this, right-click the solution and select *Deploy Business Configuration*.

> **i** Note
>
> When you use the *Deploy Business Configuration* function, a dialog box opens. You have to select one of the following options:
>
> - Deploy all business configuration content for your solution
> - Deploy only the business configuration content for your solution that is assigned to the business options you have selected in the business adaptation catalog
>
> This allows you to test both partial and complete activation of your business configuration content in the development environment.

Related Information

Business Configuration Quick Guide [page 362]

8.2.3.4 Create a BC Set Using an SAP BCO

Overview

You can create a business configuration set (BC set) using an SAP business configuration object (BCO) to configure the behavior of standard business processes in the SAP cloud solution. If you create an

implementation of an enhancement option which is for single use only, you can also define a BC set using an SAP BCO so that you can enhance the filter values.

> **i Note**
>
> - You can only create a BC set using an SAP BCO if you are developing the solution on your development tenant. For information, see Lifecycle Management of Customer-Specific Solutions on Your Development Tenant [page 155].
> - You cannot create BC sets using SAP BCOs in a solution template.

You can also create a BC set using a business configuration object that you have created in the solution. For more information, see Create a BC Set Using a Custom BCO [page 387].

Prerequisites

- You have created and opened a solution in the studio. For more information, see Create a Customer-Specific Solution [page 177].
- You have permission to access the business configuration objects (BCO) that are released with the SAP public solution model (PSM).

Procedure

1. In the *Solution Explorer*, do the following:
   1. Click the *Add New Item* button.
      The *Add New Item dialog* opens.
   2. Select *SAP* > *Business Configuration* > *Business Configuration Set* and click *Add*.
      The *Business Configuration Wizard* opens.
2. In the *Select a BCO Type and Create a BC Set* step, do the following:
   1. Under *Business Configuration Object Type*, select *Use SAP Business Configuration Object*.
   2. Under *Basic Information*, enter a name and a description.
   3. Under *Business Configuration Object*, in the search field, type the name of the SAP BCO you want to use or use the *Search* function to search for a BCO by deployment unit, namespace, or name.

      > **i Note**
      >
      > You can only use BCOs that are released with the SAP public solution model (PSM).

   Under *Business Configuration Object*, the system displays the deployment unit, the namespace, whether a fine-tuning activity is available, and a description of the BCO.
3. In the *Define the Values of Your Business Configuration Set* step, specify the field values of your BC set by doing the following:
   1. In the BCO hierarchy view, select the root node.
      If the BCO only has one node, by default, the hierarchy is hidden.
   2. In the field columns, specify values for at least all mandatory fields.
      An asterisk (*) is used to indicate mandatory fields. Field values must comply with the data type of the field. You can display information about a field by clicking the field name.

3. Repeat the steps above for each node.

4. In the *Review* step, check that the information you have entered is correct and that the namespace of your solution (`/BY*/`) is added to the key field; then click *Finish*.
   A new item with the file extension `.bcc` appears in the *Solution Explorer* under the project node. If you access the values you defined for the BCO using the scripting language, for example, you create a query, the key fields might contain the solution namepace as a prefix, which is also displayed in the *Review* step of the *Business Configuration Set Wizard*.
5. In the *Solution Explorer*, right-click your business configuration set and select *Activate*.
   The system generates the required business configuration object(s) and data type(s).
6. In the *Business Configuration Wizard*, in the *Create Business Options and Assign Solution Content* step, assign your BC set to a business option by using the arrow buttons or by means of drag-and-drop.
   For more information about assigning solution content to business options, see Create a Business Topic and Business Options [page 377].
7. In the *Solution Explorer*, right-click your solution and select *Deploy Business Configuration* to be able to use your business configuration content in a solution.
   The system deploys the business configuration content you created and makes the field values available in the development environment.

> **i** Note
>
> When you use the *Deploy Business Configuration* function, a dialog box opens. You have to select one of the following options:
>
> - Deploy all business configuration content for your solution
> - Deploy only the business configuration content for your solution that is assigned to the business options you have selected in the business adaptation catalog
>
> This allows you to test both partial and complete activation of your business configuration content in the development environment.

Result

You have created a BC set using an SAP BCO and assigned it to a business option.

Related Information

Business Configuration Quick Guide [page 362]

8.2.3.5 Create a Business Configuration Object

Overview

You can create a business configuration object (BCO) and use it as the basis of a business configuration set (BC set).

Prerequisites

You have created and opened a solution in the studio. For more information, see Create a Customer-Specific Solution [page 177].

Procedure

1. In the *Solution Explorer*, do the following:
   1. Click the *Add New Item* button.
      The *Add New Item dialog* opens.
   2. Select *SAP* > *Business Configuration* > *Business Configuration Object* and click *Add*.
      The *Business Configuration Wizard* opens.
2. In the *Create Business Configuration Object* step, do the following:
   - Under *Basic Information*, enter a name and description.
   - Optional: Select *Create Code List Data Type* if you want the system to create a code list data type that can be used, for example, in business objects and business objects extensions.
     In this case, only one key field is allowed.

     > **i Note**
     >
     > If you want to use a static code list in your solution, you can create a code list data type. For more information, see Create a Code List Data Type [page 472].
3. In the *Define Fields of the Business Configuration Object* step, for each field you define, enter the following data and click *Next*:
   - **Field name**
   - **Field description**
   - **Field type**
     Possible values are:
     - Key: key field of the business configuration object
     - Attribute: language-independent attribute of the business configuration object
     - Description: language-dependent attribute of the business configuration object
   - **Data type**
     For more information about the supported data types and default values, see Data Types [page 92].
   - **Data subtype**
     If you define a field of data type *Code*, you must select a data subtype.

- **Length**
- **Decimal**
- **Mandatory**

4. In the *Review* step, do the following:
   1. Check that the information is correct:

Name	[name of the business configuration object]
Description	[description of the business configuration object]
Data Type*	[name of the business configuration object]Code

> **i** Note
>
> * The data type is displayed if you have selected the *Create Code List Data Type* option in step 2.

   2. Click *Finish*.
      A new item with the file extension `.bco` appears in the *Solution Explorer* under the *Business Configuration* node.

5. Activate the business configuration object by right-clicking the `.bco` file and selecting *Activate*.

Result

You have created a BCO that you can now use as the basis of a business configuration set (BC set). You can reference the values of this BCO in more than one BC set. For more information, see Create a BC Set Using a Custom BCO [page 387].

Related Information

Business Configuration Quick Guide [page 362]
Create a BC Set Using an SAP BCO [page 383]

8.2.3.6 Create a BC Set Using a Custom BCO

Overview

You can create a business configuration set (BC set) using a business configuration object (BCO) that you have defined in your solution.

You can also create a BC set using an SAP BCO. For more information, see Create a BC Set Using an SAP BCO [page 383].

Prerequisites

You have created and activated a business configuration object in your solution. For more information about creating a BCO, see Create a Business Configuration Object [page 386].

Procedure

1. In the *Solution Explorer*, do the following:
   1. Click the *Add New Item* button.
      The *Add New Item dialog* opens.
   2. Select *SAP* > *Business Configuration* > *Business Configuration Set* and click *Add*.
      The *Business Configuration Wizard* opens.

      > → Tip
      >
      > You can also right-click your custom BCO and select *Create Business Configuration Set*.
2. In the *Select a BCO Type and Create a BC Set* step, do the following:
   1. Under *Business Configuration Object Type*, select *Use Business Configuration Object Defined in Solution*.
   2. Under *Basic Information*, enter a name and a description.
   3. Under *Business Configuration Object*, in *Name*, select the business configuration object on which you want to base your BC set; then click *Next*.

      > i Note
      >
      > You can only use BCOs that you created in the solution.
3. In the *Define the Values of Your Business Configuration Set* step, specify the field values of your BC set.

   > i Note
   >
   > - The field values must comply with the data type of the field. You can display information about a field by clicking the field name.
   > - As a partner, you cannot enter values in the `Key [Code]*` field that start with “Z”. This namespace is reserved for customers.
   > - When you, as a customer, enter values in the `Key [Code]*` field, these values should start with “Z”. This guarantees that your entries cannot be overwritten by content delivered by a partner solution. If, however, your business requires that you deviate from this recommendation, you should inform the SAP partner to avoid clashes of content.

   Under *Business Configuration Set Field Values*, you can select *Values can be added in Fine-Tuning*; click *Next*.
4. In the *Review* step, check that the information you have entered is correct and then click *Finish*.
   A new item with the file extension `.bcc` appears in the *Solution Explorer* under the project node.
5. In the *Solution Explorer*, right-click the `.bcc` file and select *Activate*.
6. In the *Business Configuration Wizard*, in the *Create Business Options and Assign Solution Content* step, assign your BC set to a business option by using the arrow buttons or by means of drag-and-drop.

For more information about assigning solution content to business options, see Create a Business Topic and Business Options [page 377].

7. In the *Solution Explorer*, right-click your solution and select *Deploy Business Configuration* to be able to use your business configuration content in a solution.
   The system deploys the business configuration content you created and makes the field values available in the development environment.

> **i Note**
>
> When you use the *Deploy Business Configuration* function, a dialog box opens. You have to select one of the following options:
>
> - Deploy all business configuration content for your solution
> - Deploy only the business configuration content for your solution that is assigned to the business options you have selected in the business adaptation catalog
>
> This allows you to test both partial and complete activation of your business configuration content in the development environment.

Result

You have created a BC set based on a BCO that is defined in your solution and assigned it to a business option.

Related Information

Business Configuration Quick Guide [page 362]
Create a BC Set Using an SAP BCO [page 383]

8.2.3.7 Create a Business Configuration View

Overview

You can create a business configuration view (BC view) for a business configuration set (BC set) to allow administrators to add new values during fine-tuning or to delete the values of a BC set. You create the view in the studio and you model the view in the UI designer.

> **i Note**
>
> You can only create a BC view for a business configuration set (BC set) that was created using a custom business configuration object (BCO). You cannot create a BC view for an SAP BCO.

Prerequisites

- You have created a custom business configuration object (BCO). For information, see Create a Business Configuration Object [page 386]. When you create your BCO, if you want the system to create a code list, only one key field is allowed.
- To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the *Business Configuration* work center. For more information, see User Setup Quick Guide [page 33].

Procedure

1. **In the studio, create a business configuration view by doing the following:**
   1. In the *Solution Explorer*, activate the BCO for which you want to create the view by right-clicking the `.bco` file ( ) and selecting *Activate*.
   2. In the *Properties* window, make sure that the activation status of the BCO is *Runtime objects up to date*.
   3. Right-click the `.bco` file and select *Create Business Configuration View*.
      The *Create Business Configuration View* dialog box opens.
   4. Enter a name for the view, make sure that your BCO is selected, and then click *OK*.
      A new item with the file extension `.QA.uicomponent` ( ) appears in the *Solution Explorer*.
   5. In the *Solution Explorer*, open the BC view by right-clicking the `.QA.uicomponent` file and selecting *Open in UI Designer*.
      The view opens in the UI designer.
2. **Optional: In the UI designer, modify the BC view, for example, change the field labels:**
   1. On the *Designer* tab, select the column whose name you want to change and in the *Property Explorer*, under *Text Information*, click the *Label* field.
   2. Click the arrow button and in the *Dependent Property Editor*, in the *Overridden Text* field, enter a new name.
3. **In the UI designer, assign your BC view to the *Business Configuration* work center by doing the following:**
   1. In the *Configuration Explorer*, open the *Implementation Projects* work center view by opening the `/SAP_BYD_APPLICATION_UI/BCTools/WorkCentre/` folder. The file name of the view is `ImplementationProjects_WCView.WCVIEW.uiwocview`.
   2. In the *Extensibility Explorer*, select the `ImplementationProjects_WCView.ao.uianchor` anchor and click *Assign Component to View*.
   3. In the *Type* field, select *NavigationTarget*.
   4. In the *Component field*, select your BC view from the Repository. It is stored in the following folder: *[name of your solution]_BC]* > *SRC* > *[your BC view]*. Click *OK* and *Apply*.

      > → Tip
      >
      > You can find the name of your solution in the studio: In the *Solution Explorer*, select your solution. In the *Properties*, in the *Project File* field, there is the name of your solution.

5. Save and activate your UI and close the UI designer.

4. **In the studio, create a BC set by doing the following:**
   1. In the *Solution Explorer*, right-click your `.bco` file and select *Create Business Configuration Set*.
      The *Business Configuration Set Wizard* opens.
   2. Follow the wizard steps and make sure that:
      - Under *Business Configuration Object Type*, *Use Business Configuration Object Defined in Solution* is selected.
      - Under *Basic Information*, enter a name and description for the BC set.
      - Under *Business Configuration Object*, the BCO you created is displayed in the *Name* field.
      - You define that key users can add and delete field values during fine-tuning.
        If you do not allow the addition nor the deletion of field values, customers will not be able to make any changes on the BC view.
   3. In the *Solution Explorer*, activate the BC set by right-clicking the `.bcc` file ( ) and selecting *Activate*.
   4. In the *Properties* window, make sure that the activation status of the BC set is *Runtime objects up to date*.

   For information about creating a BC set, see Create a BC Set Using a Custom BCO [page 387].

5. **Make your BC view available in the business adaptation catalog (BAC) by doing the following:**
   1. In the *Solution Explorer*, select your solution and click *Add New Item*. Select *BAC Elements* and click *Add*.
      The *Business Configuration Wizard* opens.
   2. Follow the wizard steps and create a business option or a business topic and more than one business option.
      Make sure that you select *Visible in Fine-Tuning* and, if appropriate, *Mandatory for Go-Live* for the business option to which you want to assign the BC set that you created to make your BC view accessible in fine-tuning.
   3. In the *Solution Explorer*, right-click the `.bac` file ( ) and select *Activate* to make the business option available in the BAC.

6. **Open your BC view for testing in the cloud solution by doing the following:**
   1. Log on to the cloud solution as a business user.
   2. In the cloud solution, in the *Business Configuration* work center, open the *Implementation Projects* view.
   3. Select a project and click *Edit Project Scope*.
      The *Edit Project Scope* guided activity is displayed.
   4. Follow the guided activity and answer the scoping question for your business option with *Yes*.
      The business topic or business option you created is displayed in the *Scoping* phase under *Scoping Element*. The description and scoping questions you defined are displayed in the *Questions* phase.
      When you finish the guided activity, the activity you assigned to your business option is added to the activity list.
   5. In the *Implementation Projects* view, select the implementation project and click *Open Activity List*.
      The *Activity List* guided activity opens and displays the *Prepare* phase. *Activity List* displays all activities that are mandatory for go-live. You must add any activities that are optional manually. For information about adding optional activities, see the activity list documentation in the cloud solution.
   6. In the *Activity List*, click *Hide Phased List* and search for your activity by the activity description you defined in the studio.
   7. Select your activity and click *Open*.
      Your BC view opens and you can test the result.

> **i Note**
>
> If you have assigned more than one BC view to the same business option, a fact sheet with the links to all of the views is displayed when you click the activity.

Result

You have created a business configuration view and opened it in the cloud solution. You can now test the result.

Related Information

Business Configuration Quick Guide [page 362]

8.2.3.8 Create a Fine-Tuning Project Template

Overview

You can create a fine-tuning project template in the studio as part of a solution. Fine-tuning project templates provide sample content for a specific scenario that is defined by SAP, such as Financials Chart of Account. The fine-tuning activities you need to complete for the template are explicitly listed.

Prerequisites

You have created and opened a solution in the studio. For more information, see Create a Solution [page 177].

Procedure

1. In the *Solution Explorer*, right-click your solution and click *Add* > *New Item*.
   The *Add New Item* dialog opens.
2. Select *SAP* > *Business Configuration* > *Implementation Project Template*.
3. Enter a name and click *Add*.
   The *Create Implementation Project Template* dialog opens.
4. In the *Create Implementation Project Template* dialog, do the following:
   1. Enter a name and a description.
   2. From the *Activity Group* list, select the scenario for which you want to create a template, for example, *Financials Chart of Account*.

Information about the required scope of the template and the fine-tuning activities you need to complete is displayed under the name of the activity group (currently: for *Financials Chart of Account*).

3. Click *OK* to save the template.

   A new item with the file extension `.bct` ( ) appears in the *Solution Explorer* under your solution.

5. Right-click the `.bct` file and select *Activate*.
   The status of the template changes to *Active*.
6. Right-click the `.bct` file and select *Open*.
   The template opens in a document window.
7. Under *Scoping and Fine-Tuning*, click *Edit Project Scope*.
   The logon screen of the SAP cloud solution opens in your browser. Do the following:
8. Enter the user name and password that you entered when logging on to the repository.
   The *Edit Project Scope* guided activity is displayed. Do the following:
   1. In the *Country and Type of Business* step, click *Edit Countries* to add or remove countries and click *Edit Type of Business* to select the type of business.
   2. In the *Implementation Focus* step, select *Complete Solution* and click *Next*.
   3. In the *Scoping* step, define the required scope of your template based on the information displayed under *Activity Group* and click *Next*.

      > **Example**
      >
      > If you have created a fine-tuning project template for the *Financials Chart of Account* activity group, select *Financial Management and Accounting* > *General Ledger*.

   4. In the *Questions* step, answer the relevant scoping questions based on the information displayed under *Activity Group* and click *Next*.

      > **Example**
      >
      > Select *Financial and Management Accounting* > *General Ledger* > *General Ledger* and under *Income Statement by Function of Expense*, select the *In Scope* checkbox. Confirm the *Review Status* popup by clicking *Yes*.

   5. In the *Review* step, review your scope changes and click *Finish*.
   6. In the *Confirmation* step, click *Close*.
9. In the studio, refresh the template by clicking *Refresh* in the document window and then click *Confirm Milestone: Design Accepted*.
   In the cloud solution, the *Confirm Milestone: Design Accepted* view opens.
10. Select *Design Accepted* and click *Confirm* to confirm that you have finished defining the scope of your template. Confirm the *Design Accepted* popup by clicking *Yes*.

    > **Note**
    >
    > You cannot move on to the next phases until you have confirmed this milestone. The confirmation process can take up to 30 minutes.

11. In the studio, refresh the template by clicking *Refresh* in the document window .
    The links to the individual activities that you need to complete are activated.
12. Complete the activity list by clicking the links to open each activity in the fine-tune phase of the implementation project.

Result

You have created a fine-tuning project template in the studio and predefined the fine-tuning activities for a specific scenario. You can now complete your implementation project template. For more information, see Business Configuration Quick Guide [page 362].

Related Information

Complete the Activity List [page 396]

8.2.3.9 Create a Tax Decision Tree

You can create a tax decision tree in the studio to automatically determine a tax event for a country that does not have an SAP country version in SAP Business ByDesign.

Prerequisite

You have created and activated the tax content of the country for which you want to create the tax decision tree. For more information, see Create Tax Content for Non-Localized Countries in the SAP Cloud Applications Studio.

Procedure

1. In the *Solution Explorer*, do the following:
    1. Select the *Withholding Tax* checkbox if you want to create the decision tree for withholding tax determination.
    2. Click *Add New Item*. The *Add New Item* dialog opens.
    3. Select *Tax Decision Tree* and click *Add*. The *Tax Decision Tree Wizard* opens.
2. In the *Create Decision Tree* step, do the following:
    1. Add a *Name* (2 characters only) and a *Description*.
    2. Select a country from the dropdown list and specify a date from which the decision tree is valid.
    3. Click *Next* to go to the next step.
3. In the *Define the Decision Tree Structure* step, do the following:
    1. Click *Add Question*. The system creates a root node where you can add your first question.
    2. Double-click the root node to define the question type. The *Question Form* dialog opens where you can choose from three different parameters: *Question*, *Result*, and *Error*.
    3. If you select *Question* and click *OK*, the system gives you an option to enter a *Question*, select a *Test Parameter*, *Comparison Parameters*, and *Comparison Value*.

4. Specify the question and select the required test parameter, comparison parameters, and comparison value.

> **i Note**
>
> The system provides you with a list of parameters that will help you create your decision tree:
>
> **VAT Decision Tree test parameter set**

Test Parameter	Context Parameters	Comparison Value
Direction Indicator	No contexts	1 – Sale, 2 – Purchase
Product Taxability	No contexts	1– Good/Material, 3 – Asset Individual Material, 2 – Service, 6 – Entitlement, 4– Warranty
Country	Partner Role, values – SF (Ship From), ST (Ship To), CF (Contract From), CT (Contract To)	Country code (Automatically determined)
ServicePointTaxableIndicator	Tax Type, Taxable Country, Taxable Region	X (true)
Tax Exemption Reason Code	Tax Type, Taxable Country, Taxable Region	Tax exemption reason code value
Business Partner Taxability	Partner role (similar to Country test parameter)	100 – Company

> **Withholding Tax Decision Tree test parameter set**

Test Parameter	Context Parameters	Comparison Value
ProductTaxTypeIndicator	Taxable Country, Tax Type, Taxable Region	X (true)
PartnerTaxGroup	Taxable Country, Partner Role [values -SF (Ship From), ST (Ship To), CF (Contract From), CT (Contract To)], Tax Type, Taxable Region	Withholding Tax Rate Type value for the corresponding tax type selected in the context parameter
IncomeType	Taxable Country, Tax Type	Income type value for the corresponding tax type selected in the context parameter

5. Click *OK*.
   The system adds the question to the tree.

6. Click *Add Question* again. The system creates two nodes: *True* and *False*. Double-click *True* to add a question as above. Similarly add a question to *False*. Continue the steps till you reach a result or an error and click *OK* to finalize your decision tree.
7. Click *Next* to go to the *Review* step.

4. In the *Review* step, check that the information you have entered is correct and then click *Finish*.
   A new item with the file extension .bccxtax appears in the *Solution Explorer* under the project node.
5. In the *Solution Explorer*, right-click the .bccxtax file and select *Activate*.
6. In the *Solution Explorer*, right-click your solution and select *Deploy Business Configuration* to be able to use your decision tree in a solution.
   The system deploys the decision tree content you have created and makes the decision values available in the Business ByDesign system.

Result

You have created a tax decision tree, which you can use to determine the tax event when you process an invoicing document in the SAP Business ByDesign system.

8.2.3.10 Complete the Activity List

Overview

You can match business requirements to the capabilities of your solution by completing an activity list of the implementation and project management activities that customers need to perform before the solution can go live. You do this in the *Prepare* phase, the *Fine-Tune* phase, and the *Integrate and Extend* phase of an implementation project.

The system generates the activity list based on the scope defined for the implementation project template. The activity list is divided into separate phases and includes all mandatory implementation and project management activities that customers need to complete before the solution can go live. You can add optional activities to the activity list and complete them; most optional activities relate to the *Fine-Tune* phase but may also relate to other phases. In the *Prepare* phase and the *Fine-Tune* phase, you must confirm milestones that allow you to track the status of your implementation project template.

Prerequisites

You have created and activated a fine-tuning project template in the studio and you have defined the scope of this template in the SAP cloud solution. For more information, see Create a Fine-Tuning Template [page 392].

Procedure

1. Open the activity list to complete it for your implementation project template by doing the following:
    1. In the studio, in the *Solution Explorer*, right-click the `.bct` file and select *Open Project Overview*.
    The logon screen of the SAP cloud solution opens in your browser.
    2. Enter the user name and password that you entered when logging on to the repository.
    The *Project Overview* view opens.
    3. Check that the description is correct and that the name in the *Title* field is the name of your implementation project template with your project namespace as the prefix.
    4. Click *Open Activity List*.
    The *Activity List* guided activity opens and displays the *Prepare* phase.

    > ⚠ Caution
    >
    > To ensure that an implementation project template and its solution remain consistent, always open an implementation project template in the *Project Overview* view by using the *Open Project Overview* function in the studio.

2. In the *Prepare* phase, do the following:
    1. In the *Show* filter, select *My Activities in Project* to find all activities that are assigned to you.
    2. Complete the activities by doing the following:
        1. To display which activities have to be completed first, in the *Prerequisites Exist* column, click *Yes*.
        A dialog box opens and lists the prerequisite activities.
        2. Select the first activity and click *Open*.
        3. Read the instructions for the activity and complete the activity as described.
        4. Save your changes and close the activity.
        5. Repeat the above steps for each activity.

        > ℹ Note
        >
        > Some activities contain data that cannot be copied to another system; therefore, they need to be repeated in other systems. You can identify these activities in the *Repetition Required* column.
        >
        > If you complete an activity in the development environment that requires repetition, the customer must repeat this activity in the production system. Therefore, we recommend that you provide documentation for these activities in your Solution Documentation. For more information, see Documentation Types in the SAP Solution (ESS_CON_PDI_MainDocTypes.xml). and Templates Quick Guide (TemplatesQuickGuide.xml)..

    3. On the *Activity List* screen, change the status of each completed activity to *Closed*. To do this, select each activity, click *Change Status*, and then select *Closed*.

        > ℹ Note
        >
        > The *Activity List* screen indicates whether an activity is open, in process, or closed.
        >
        > At the top of the screen, you can view the overall progress for each phase of your implementation project template. At the bottom of the screen, you can view the total number of activities in the activity list for the selected phase and the numbers of open, in process, and closed activities.

    4. Confirm the *Confirm Milestone: Design Accepted* milestone.

3. In the *Fine-Tune* phase, do the following:
    1. Complete the activities as described for the *Prepare* phase.

        > **i Note**
        >
        > - Additional Information
        >   Some activities in this phase have additional information to assist customers in completing them. You can access the additional information either directly in the *Help Center* of the relevant activity under *Help Resources* by clicking *Help Center* in the *Infobar*.
        > - Mandatory and Optional Activities
        >   By default, the activity list contains only mandatory activities. In the *Fine-Tune* phase and the *Integrate and Extend* phase, you can add optional activities to the activity list.
        > - Translate Configuration Settings
        >   You can only provide your configuration settings in fine-tuning activities in one language.
        > - Restore Default Fine-Tuning Settings
        >   To avoid causing inconsistencies, you cannot delete standard fine-tuning settings.

    2. Confirm the *Confirm Milestone: Solution Accepted* milestones.
4. In the *Integrate and Extend* phase, complete the activities as described for the *Prepare* phase and enter basic data.

    > **i Note**
    >
    > The activities you need to complete in the *Integrate and Extend* phase depend on the scope of your implementation project template.

Result

You have completed the activity list. You can now complete your fine-tuning project template in the studio. For more information, see Business Configuration Quick Guide [page 362].

8.3 Business Objects

8.3.1 Business Objects Quick Guide

In the studio, you can create your own business object and define its nodes and relevant elements in the business object definition. After you have defined your business object, you can generate the screens for your business object.

Availability

> **⚠ Caution**
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud

solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.

For details, see Feature Matrix [page 27].

Business and Technical Background

Scripting Language

The scripting language is used to define business objects and business object extensions as well as to implement the business logic for cloud partner solutions. This lightweight language is easy to learn and to use with its focused set of features.

For more information, see here [page 189].

Syntax for Business Object Definition

You use specific keywords and functions to describe the structure and interface of a business object in the studio.

For more information, see here [page 191].

Screens

In the *Create Screens* dialog, you select the screens you want to use for your solution.

For more information see Screen Types [page 408].

Object-Based Navigation and Thing-Based Navigation

In the studio, you can choose between two different screen scenarios.

For more information, see Object-Based Navigation and Thing-Based Navigation [page 412].

Queries

A query is a service operation that can be used for search and information retrieval operations. It is defined at a business object node and has selection parameters. The result structure is defined by the business object node the query is specified for; however, it may also include elements from different business object nodes.

For more information, see here [page 406].

Actions, Events and Validations

In the studio, script files are used to implement the business logic for actions, events or validations of business object nodes.

For more information, see here [page 403].

Syntax for Implementation of Business Logic

Methods and keywords enable you to implement the business logic for your solution in your script files in the studio.

For more information, see here [page 235].

Reuse Libraries and Built-In Functions

The built-in functions and reuse libraries of the scripting language extend the scope and functions that you use to implement the business logic of your solutions.

For more information, see the following documentation:

- Built-In Functions [page 320]
- Basic Reuse Libraries [page 274]
- Reuse Libraries for Business Areas [page 308]

Introduction to Business Object Modeling

The SAP cloud solutions are built using the concept of business objects to model the business environment. For more information, see here [page 81].

For an example on how to build a solution, see Example: Building a Solution [page 99].

Public Solution Model

In the studio, you can explore SAP content and PSM documentation using the *Repository Explorer* tool window. For information, see Repository Explorer [page 49].

Tasks

Create a Business Object

1. In the *My Solutions* tool window, open your solution.
2. In the *Solution Explorer*, right-click the project in your solution and select *Add* > *New Item*.
3. In the *Add New Item* dialog box, select the *Business Object* template, enter a name for your business object, and click *Add*.

   > **i** Note
   >
   > If you use a template to create a new project item, do not change or delete the extension of the item name. For example, do not delete the `.bo` file extension of the business object file.

   A new item with the file extension `.bo` appears in the *Solution Explorer*.
   A document window opens that displays your business object file in a code editor.

Enter a Business Object Definition

In the code editor of the studio, you define your business object by describing the namespace and by adding business object elements. To describe the structure and interface of the business object, you use specific keywords and functions.

For more information on the syntax of business objects, see Syntax for Business Object Definitions [page 191].

For more information about entering a business object definition, see Enter a Business Object Definition [page 418].

Add a Label and Tooltip to a Business Object

You can define a label and a tooltip for a business object field and display them on a screen.

For more information, see here [page 419].

Activate a Business Object

Once you have entered the business object definition in the code editor and saved it, you need to activate the business object. This ensures that the runtime objects are generated that are required for the business object to function.

For more information about activating a business object, see Activate a Business Object [page 420].

Define the Business Logic for a Business Object

To define the business logic for your business object in the studio, you use event, action and validation script files.

For more information, see here [page 428].

Generate Screens for a Business Object

In the studio, you can generate screens in the SAP cloud solution based on your business object definitions, for example, work centers, floorplans, and fact sheets.

For more information, see here [page 429].

Create a Query

In the studio, you can define queries that allow you to access and combine the fields defined in different nodes or business objects in a single query and to combine data from different business object nodes. You can also use a query to make a custom business object available in the enterprise search.

For more information about creating a query, see here [page 421].

Define Access Control

In the studio, you can implement instance-based access control for a business object you created by referencing an SAP business object for which an access context is defined.

For more information about defining access control, see here [page 424].

Assign Access Rights

If you have implemented instance-based access control for a business object you created in the studio, you can restrict read and write access in the SAP cloud solution on the basis of a work center view's access context.

For more information about assigning access rights, see here [page 427].

Create an OWL Excel Export

Object Work Lists (OWL) generated for custom business objects allow you to export the OWL data to a Microsoft Excel file. The file is based on a standard template, which you cannot change. However, you can format the generated spreadsheet as required.

For more information, see here [page 431].

Enable Change History

You can use an embedded component to ensure that a change history is written for all changes made to your custom business object. After adding the embedded component to your business object, you can display the change history in the SAP cloud solution on the *Changes* tab of your work center. All changes such as add, create, and delete are displayed in a list.

For more information, see here [page 435].

Related Information

Maintenance of Business Objects [page 414]

8.3.2 Business and Technical Background

8.3.2.1 Actions, Events, and Validations

In the studio, script files are used to implement the business logic for actions, events or validations of business object nodes.

Depending on whether you are implementing the business logic for business objects or the business logic for business object extensions, you can create the following script files:

Business Object	Business Object Extension
Action	—
`AfterModify` event	`AfterModify` event
`BeforeSave` event	`BeforeSave` event
`AfterLoading` event	`AfterLoading` event
`OnSave` validation	`OnSave` validation

Business Object	Business Object Extension
—	Action validation
`OnDelete` event	—

Features

Actions

An action is an element of a business object node hat describes an operation performed on that node, for example, setting a status. The action operates on a set of node elements or instances that have been created or changed. Action script files are provided if you have added actions to your business object definition.

Event

An event is used to compute data that can be derived from other node elements, for example, totals of items and statuses, or from dependent objects, for example, `AttachmentFolder` or `TextCollection`.

The system executes an event implemented for custom business objects in ascending order from bottom to top, that is, from subnode to root node. In case of events implemented for extended business it is the reverse: the system executes an event in descending order from top to bottom, that is, from root node to subnode.

The following events are provided by the system:

- `AfterModify`
  This event determines the data of a node when the data has been changed. This is especially useful for derived data that needs to be updated immediately.
- `BeforeSave`
  This event determines the current data of a node.

  > **i Note**
  >
  > Please note that if a `BeforeSave` script file has modified a business object node, the system doesn't execute the code of an `AfterModify` script file for this same node anymore.

- `AfterLoading`
  You can implement this event to modify **transient fields** (only available for business objects) of the current node of a business object. The system executes the `AfterLoading` event when it reads a node instance of a business object from the data base.
  You **cannot** use an `AfterLoading` event to do the following:
  - Read and modify a transient field of another node
  - Modify a persistent field
  - Create or delete a node
  - Call an action

  > **i Note**
  >
  > The system doesn't execute an `AfterLoading` event for a node instance that is being initially created. To calculate a transient field at this point in time, you can implement an `AfterModify` event.

Validations

A validation is used to check if particular conditions are fulfilled, such as the conditions for saving a business object node. Validations can only read data and cannot change any values. The result of a validation must always evaluate to a Boolean value, that is, to `true` or `false`. The system executes validations in no particular order. It executes all of the validations, that is, even if a validation already returned false, the remaining validations are executed.

You can create the following validation script files:

- `OnSave` validation
  You can use the `OnSave` validation to check whether a business object can be saved.

  > **i Note**
  >
  > Please note that the system always executes the `OnSave` validation after having executed the `BeforeSave` event.

  > **Example**
  >
  > A validation that checks whether the string “0243” is contained in the `elemData` field:
  >
  > ```
  > if (this.elemData.Contains("0243")) {
  >   return(true);
  > }
  > else {
  >   return(false);
  > }
  > ```

  For another example, see Example: Validate on Save [page 138].
- Action validation (available for business object extensions only)
  You can use an action validation to check whether an action of an SAP business object can be executed. As a prerequisite, the action must be released in the public solution model (PSM). The system executes the validation before it calls the corresponding action. The return value determines whether the action is executed or not. If the action contains parameters, you can use them in the script file of the validation. You can raise messages defined in the business object extension if the action cannot be executed.

  > **i Note**
  >
  > You can create action validation script files only in solutions or solution templates that have been created with the `1302/1211` compiler version or higher. You can find the compiler version in the *Properties* window of your solution.

Mass Enablement

You can specify whether a script file is to be executed for multiple instances of business object nodes, which is referred to as mass enabling.

If you work with mass-enabled script files, consider the following:

- To mass enable a script file, you select the *Mass Enable* checkbox in the *Create Script Files* dialog box. For more information, see Define the Business Logic for a Business Object [page 428].
- If you want to modify a mass-enabled script file so that it can only be executed for single instances of business object nodes, you need to delete the script file first and then re-create it.

Maintenance Mode

You can delete a script file when your solution is in maintenance mode. For more information, see Maintenance Mode [page 185].

> **i Note**
>
> If you use the *this* keyword in the coding of a mass-enabled script file, *this* represents a collection of business object node references.

Performance Checks

This feature provides you with suggestions on how to avoid dumps and improve performance of ABSL scripts, business objects, business object extensions, mass data runs, and the solution itself. Right-click on any .absl, .bo, .xbo, and .run file in the solution and click *Run Performance Checks*.

When you run a check on scripts, business objects, business object extensions, and mass data runs, the system displays the list of tips that will enhance your system performance, and allows you to directly work on the script without moving away from the screen. When you run a check on the solution, due to the large volume of objects involved, the system runs a background job and sends you an e-mail with the list of tips and the particular lines where you can implement the improvements.

8.3.2.2 Queries

Overview

A query is a service operation that can be used for search and information retrieval operations. It is defined at a business object node and has selection parameters. The result structure is defined by the business object node the query is specified for; however, it may also include elements from different business object nodes.

You can use queries to display business object data directly in an object worklist (OWL) using the user interface designer (UI Designer) or to access the data using the scripting language. For more information about the usage in UI designer, see Model an Object Work List [page 703] and Configure a Query [page 741].

Query Types

Default Query

By default, each business object node of a custom business object has a query called `QueryByElements`. The query contains all elements of the node as selection parameters and the query result contains instances of this node. In general, the `QueryByElements` query has a linear dependency on the number of instances in the business object node (t = O(n), where n is the number of business object instances in the database).

The `QueryByElements` query is a simple query and does not provide full-text indexing. We therefore recommend that you use the default query when:

- The expected number of node instances or collections is small, for example, for an object with configuration data, or when the execution frequency is low.

- The selection parameter list contains an equal condition on an element that is the alternative key of the node. An alternative key is supported by an index so the runtime dependency is t = O(log n).

> ⚠ Caution
>
> If you define an association in your business object as relevant for access control by using the `RelevantForAccessControl` annotation and you use the default `QueryByElements` query, error messages related to role-based access management (RBAM) might occur if you use the query to control access. These RBAM error messages are suppressed when you create a complex query in your solution.

Complex Query

A complex query is a custom query that allows you to access and combine data from different business object nodes or business objects in a single query. Complex queries support full-text indexing for all query elements (t = O(log n). We therefore recommend that you define a complex query when a large number of records is expected in the node.

In the studio, you can create queries using the *Query Wizard*. For information, see Create a Query [page 421]. When you define a query, you must specify which business object fields to include in the query. You can also specify which elements you want to:

- Include as search parameters
- Use in text search
- Display in the query result

If an [1,n] association is defined for an element, you cannot use the field in the query result. We therefore recommend that you select the most detailed node as the parent node. For more information about associations in business object definitions, see Association [page 196].

> → Tip
>
> If you want to implement instance-based access control for the results displayed in an object worklist (OWL), you must create a complex query. You cannot create a default query for this scenario. For information about defining access control, see Define Access Control [page 424].

Predefined Query of SAP Business Objects

Most of the business objects provided by SAP have predefined queries which you can use in the same way as a default query or a custom query. However, you cannot use sorting and paging options for a predefined query.

Executing Queries

- **Execute Function in Script Files**
  In script files, you typically use the `Execute` function which allows to pass selection parameters to the query and returns a collection of node instances that match the selection. A query reads the data of a node

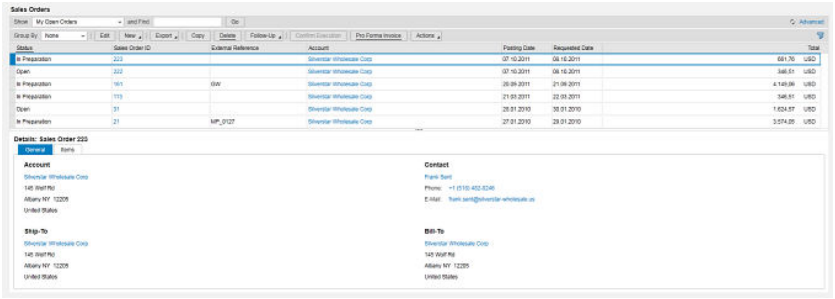
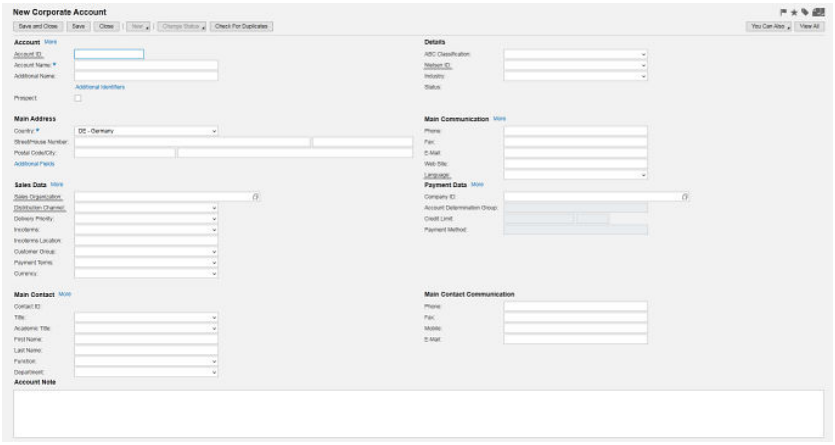
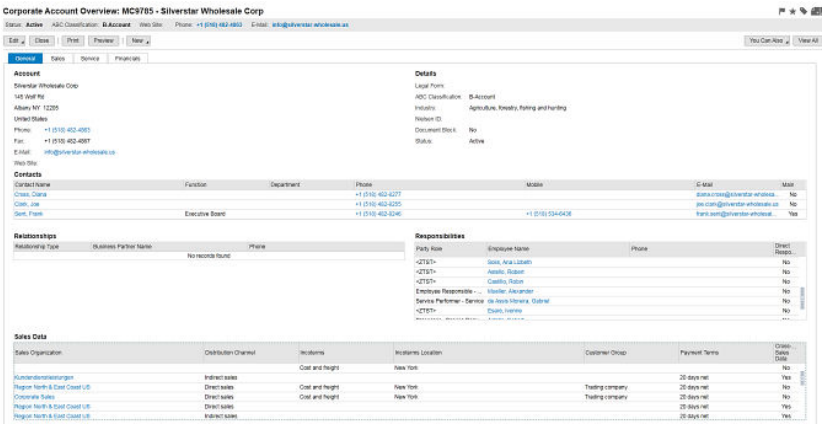
directly from the database which means that unsaved data that has been created or modified in the current session or a session running in parallel is not taken into account.
If a query returns a collection of nodes, the collection is not sorted by default. However, if you have defined a query on a custom business object, you can specify that the result is to be sorted by specified elements. Another option is to use the `Sort Collection` functions (see Sort Collection [page 271]). Please note that sorting a collection in memory is slower than sorting performed by a query that is executed on the database.
If you need to process only a small part of the result and the number of nodes matching the selection criteria is quite high, you can optimize the amount of data that needs to be read from the database by using a sort option. For example, if you want to find the last document saved by a particular user and the search criterion is "Changed by the current user", you can use the sort option "Descending by last changed date time" and specify the maximum number of rows to be returned, for example "1".
Obtaining business object nodes by using the `Execute` function gives you the most flexibility if you want to further process the query result. You can then read or modify the data of the result, execute actions or retrieve further data using associations. However, since the returned nodes reside in the buffer of the runtime, they need to be considered whenever the runtime checks the buffer for modification. Thus, if you expect a large result set and your solution does not need to modify the returned business object nodes, we recommend to query for business object node data only. This can be achieve by using the `ExecuteDataOnly` function. This function offers the same parameters as the `Execute` function but the query result is just a collection of node data instead of node instances. Therefore, you cannot modify the data returned and you cannot directly execute actions or use associations. However, the buffer of business objects nodes remains unchanged which increases performance and avoids overriding instance that are already in the buffer.
If you do not even need to access the node data but are just interested in the number of entries that exist on the database, you can use the `Count` function of the query. Count does not retrieve any data from the database which means it is faster than performing the `Count` function on the collection that has been returned by the `Execute` function or the `ExecuteDataOnly` function.

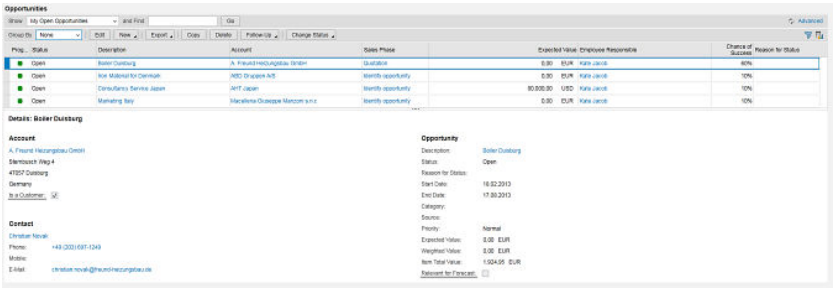
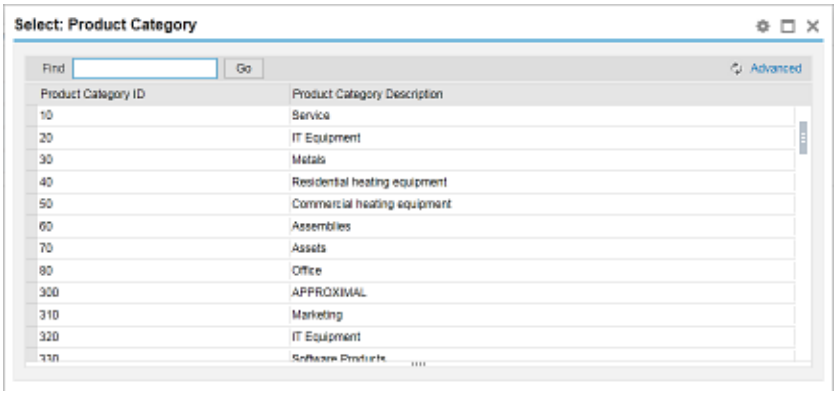
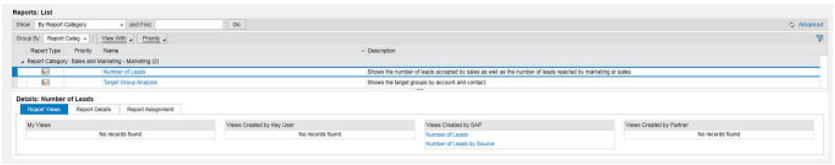
- **Execute a Query to Explore a Business Object**
  The studio enables you to execute queries for SAP business objects and custom business objects to help you better understand the business object data model. For SAP business objects, you access this function in the *Repository Explorer*, for custom business objects, you call up this function in the *Solution Explorer*. For more information, see Execute a Query to Explore a Business Object [page 52].

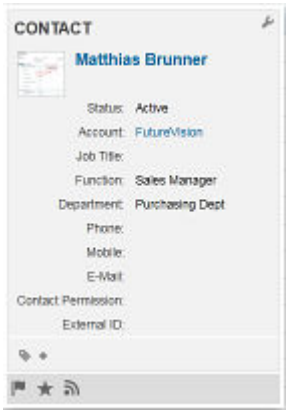
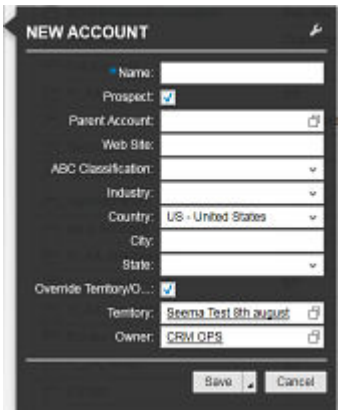
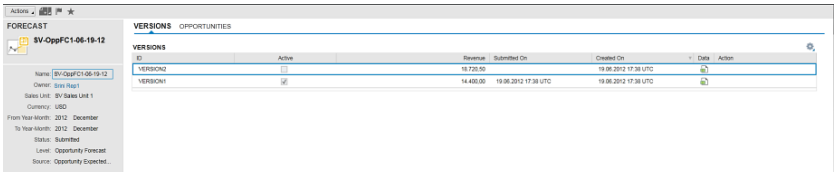
8.3.2.3 Screen Types

The studio enables you to create screens for your business object. In the *Create Screens* dialog, you are prompted to select the screen types you want to use. For information on how to create screens, see Generate Screens for a Business Object [page 429].

You can select the following screen types:

Screen Type	Description	Example
Object Instance Floorplan (OIF)	A floorplan that allows business users to create, view, and edit a business object. Typically, the OIF shows multiple tab pages. You can use this floorplan to model an application interface that allows the user to create, delete, display, and change the attributes and associations of business objects.	 Object Instance Floorplan
Quick Activity Floorplan (QAF)	A floorplan that allows business users to perform a specific task quickly. This can be self-contained or a short subtask within the context of a larger task (for example, a simple *Create* operation).	 Quick Activity Floorplan
Fact Sheet (FS)	A sheet that displays the most important properties of a business object on one page.	 Fact Sheet

Screen Type	Description	Example
Object Work List (OWL)	A list pane that is always embedded in a work center view. The OWL contains business objects such as purchase orders or opportunities. Business users can, for example, search for, display, and edit business objects as well as trigger follow-on processes.	 Object Work List
Object Value Selector (OVS)	A user interface pattern that supports business users when they select values for an input field. An OVS helps business users to select one item from a table of business object instances. If you choose *Object Value Selector (OVS)*, a separate dialog box appears in which you can select the business object elements for the OVS.	 Object Value Selector
Work Center View (WOC-VIEW)	Part of a work center, for example, the *Overview* or *Reports* view.	 Work Center View

Screen Type	Description	Example
Quick View (QV), not available for SAP Business By-Design 	A slide-in window, which allows users to edit data without navigating away from the UI currently displayed.	 Quick View
Quick Create (QC), not available for SAP Business By-Design 	A preview of detailed information on a thing that a user can see without navigating away from the UI currently displayed. Data shown in a QC cannot be edited.	 Quick Create
Thing Inspector (TI), not available for SAP Business By-Design 	The TI shows all aspects of a thing and is a compact summary of all primary attributes, the tags, and the actions that can be carried out.	 Thing Inspector

Screen Type	Description	Example
Thing Type (TT), not available for SAP Business By-Design	A TT describes the general UI properties of a thing. It is used for navigating between the screens that are thing-based and for the provisioning of REST services for a thing.	

8.3.2.4 Object-Based Navigation and Thing-Based Navigation

Overview

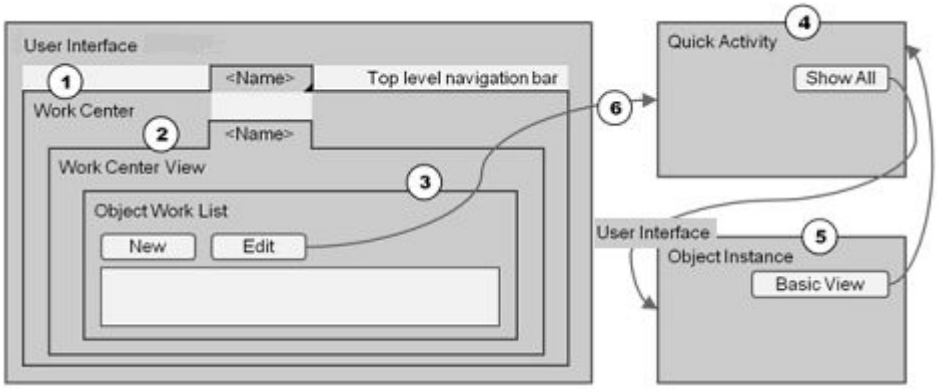
When you create the screens for your business object in the studio, you are asked to select the screen navigation you want to use. The following sections describe the two types of navigation that are available (object-based and thing-based navigation) and help you decide when to use what.

Object-Based Navigation

This is the navigation pattern used in SAP Business ByDesign.

When you use object-based navigation, you enter the application via work center and work center view to get to the object work list. Then you leave the object work list to work on quick activity or object instance floorplan screens.

The following graphic shows the various screens that you can create for object-based navigation:



Screens Using Object-Based Navigation: (1) Work Center, (2) Work Center View, (3) Object Work List, (4) Quick Activity, (5) Object Instance, (6) Object-Based Navigation

Thing-Based Navigation

This is the navigation pattern used in the other solutions. The fundamental integral part of thing-based navigation is a **thing**.

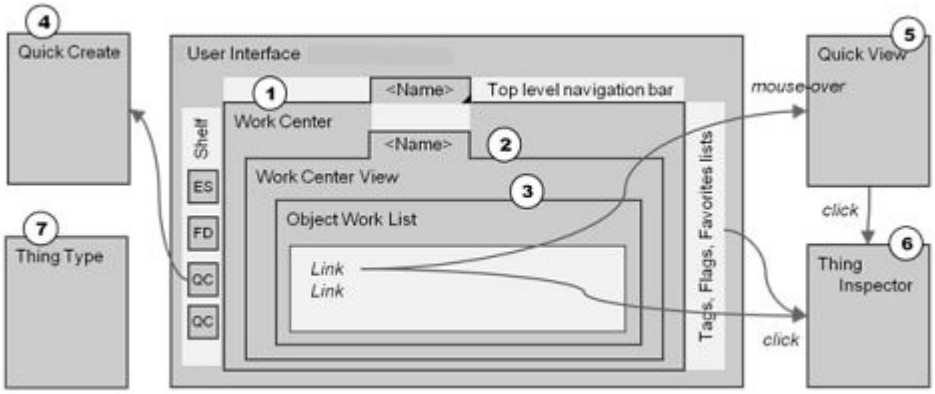
What Is a Thing?

A thing is an object a user works with, such as product or business partner. It contains content that has been collected from various applications, actions, and business objects. A thing is supposed to ensure easy consumption of that content and a consistent user experience. It is assigned to a thing type.

> ✤ Example
>
> For the thing **ABC Corporation**, the thing type would be **Business Partners**.

When you use thing-based navigation, you also navigate via a work center and a work center to an object worklist as in object based navigation. However, in thing-based navigation, you can call up quick view and quick create screens without leaving the object work list or go to a thing inspector screen.

The following graphic shows the various screens that are part of thing-based navigation:



Screens Using Thing-Based Navigation: (1) Work Center, (2) Work Center View, (3) Object Work List, (4) Quick Create (QC), (5) Quick View, (6) Thing Inspector, (7) Thing Type, ES = Enterprise Search; FD = Feeder

When Do I Use What?

Generally speaking, you select the navigation pattern that is suited best for your solution. However, there are a few restrictions that are listed in the following table:

	Object-Based Navigation	Thing-Based Navigation
Can Be Used for SAP Business ByDesign	Yes	No
Can Be Used for Other Cloud Solutions	Yes	Yes
Need to Define Outports in UI Designer	Yes	No

Related Information

Generate Screens for a Business Object [page 429]
Screen Types [page 408]

8.3.2.5 Maintenance of Business Objects

Overview

A customer-specific solution is in maintenance mode, if it has been assembled and downloaded; the solution status then is *Assembled*. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Change and Delete Restrictions

The following table gives an overview of the change and delete restrictions for business objects in a solution that is in maintenance mode. The system performs the change and deletes restrictions checks when you save the business object.

	Adding Allowed	Deletion Allowed	Renaming Allowed	Comments
Business object	Yes	No	No	
Business object node	Yes	No	No	
Business object node element	Yes	No	No	
Business object action	Yes	Yes	Yes	

	Adding Allowed	Deletion Allowed	Renaming Allowed	Comments

Business object node association	Yes	No	Yes	**i Note**
				The following changes are **not** allowed:
				• Removing the `using` clause
				`association ToBonusPlan to BonusPlan using ID -> association ToBonusPlan to BonusPlan`
				• Changing the target of an existing `using` clause
				`association ToBonusPlan to BonusPlan using ID -> association ToBonusPlan to BonusPlan using ID2`

Data type of element	No	No	No	
`AlternativeKey` annotation on existing element	No	Yes	No	
`AlternativeKey` annotation on new element	Yes	No	No	

	Adding Allowed	Deletion Allowed	Renaming Allowed	Comments

Multiplicity of node	see comments	see comments	see comments	The following changes are allowed:
				`[0,1] –> [0,n]`
				`[1,1] –> any`
				`[1,n] –> [0,n]`
				The following changes are **not** allowed:
				`[0,1] –> [1,1] or [1,n]`
				`[0,n] –> any`
				`[1,n] –> [0,1] or [1,1]`

Multiplicity of association	see comments	see comments	see comments	The following change is allowed:
				`[1,1] –> [0,1]`
				The following change is **not** allowed:
				`[0,1] –> [1,1]`

Association valuation	see comments	see comments	see comments	Adding or removing the `Valuation` keyword is not allowed.
Multiplicity of association valuation	see comments	see comments	see comments	Only the following multiplicity is allowed: [0,n]
Association valuation condition	Yes	Yes	not applicable	Changing the conditions within a valuation expression is allowed.
Messages	Yes	Yes	Yes	
Business object query	Yes	No	No	

	Adding Allowed	Deletion Allowed	Renaming Allowed	Comments
Assignment of deployment unit	Yes	Yes	Yes	**i Note** • If associations to business objects in the deployment unit are defined, changes to the assignment cause an error due to a cross-deployment unit association. • If an integration scenario has been defined, this has to be adapted.

8.3.3 Tasks

8.3.3.1 Enter a Business Object Definition

In the code editor of the studio, you define your business object by describing the namespace and by adding business object elements. To describe the structure and interface of the business object, you use specific keywords and functions.

For more information on the syntax of business objects, see Syntax for Business Object Definitions [page 191].

Prerequisites

You have opened your solution.

Procedure

1. Select the document window of your business object.
   In the code editor, the following information is displayed:
   - The `import` statement for the `AP.Common.GDT` SAP namespace that contains the standard data types:

     ```
     import AP.Common.GDT as apCommonGDT
     ```

     You can import further namespaces so that you can use SAP business objects and data types when you define elements and associations.
   - The `businessobject` keyword and the name of your business object, for example, `BonusRule`.

     > **i Note**
     >
     > The names of the `.bo` file and of the business object must be identical. Do not change the name of the business object in the coding.
2. Define the nodes of your business object and their features, such as elements, actions, and associations with other nodes.
   In the following simple business object definition, the `ID` element is specified for the root node of the `BonusRule` business object:

   ```
   import AP.Commmon.GDT as apCommonGDT;
   businessobject BonusRule {
   element ItemID : ID;
   }
   ```

   For more information about the keywords that you can use, see Syntax for Business Object Definitions [page 191].
3. In the *File* menu, click *Save*.

Result

You have defined your business object and can now activate it. For more information, see Activate a Business Object [page 420].

8.3.3.2 Add a Label and Tooltip to a Business Object

Overview

You can define a label and a tooltip for a business object field. If you then add the business object field to a screen:

- the label is displayed on the screen instead of the field name
- the tooltip is displayed when you place the cursor on the field

Procedure

1. Create a business object element with a label and a tooltip using the following syntax:

   ```
   [Label ("<label name>")] [Tooltip ("<tooltip name>")] element <element
   name> : <data type>;
   ```

   For more information, see Business Objects Quick Guide [page 398].
2. Generate screens for the business object. For more information, see here [page 429].

Result

In the *Solution Explorer* in the studio, right-click your screen and select *Preview Screen*. You are prompted to log on to the cloud solution.

The label is displayed for the business object field. The tooltip is displayed when you place the cursor on the field.

Related Information

Label (Business Object) [page 209]
Tooltip (Business Object) [page 214]

8.3.3.3 Activate a Business Object

Once you have entered the business object definition in the code editor and saved it, you need to activate the business object. This ensures that the runtime objects are generated that are required for the business object to function.

Prerequisites

You have defined your business object in the code editor. For more information, see Enter a Business Object Definition [page 418].

Procedure

In the *Solution Explorer*, right-click your business object and select *Activate*.

The content of your business object is activated.

→ Tip

How you can solve business object activation errors

If an error message appears that tells you that the business object cannot be activated, have a look at the following table. It shows possible errors and how you can solve them:

What the error may be caused by	How to solve the error
The database table that is generated from the node definition is too large.	• Delete some of the elements. • Use smaller data types. • Split up the node(s).
There has been an incompatible change in the definition of an element.	• Change the element definition back. • Clean the business object. ⚠ Caution Please be aware that data may get lost.

8.3.3.4 Create a Query

In the studio, you can define queries that allow you to access and combine the fields defined in different nodes or business objects in a single query and to combine data from different business object nodes. You can also use a query to make a custom business object available in the enterprise search.

You can use these queries to include the query data in an object worklist (OWL) and to access information from different nodes and business objects using the scripting language. You must create a query to implement instance-based access control for the results displayed in the OWL.

⚠ Caution

If you define an association in your business object as relevant for access control by using the `RelevantForAccessControl` annotation and you use the default `QueryByElements` query, error messages related to role-based access management (RBAM) might occur if you use the query to control access. These RBAM error messages are suppressed when you create a complex (custom) query in your solution.

Prerequisites

- You have created and opened a solution in the studio. For more information, see Create a Customer-Specific Solution [page 177].

- You have created and activated at least one custom business object in your solution. For information, see Business Objects Quick Guide [page 398].
- To make a custom business object available in *Enterprise Search*:
  - The custom business object must contain a description field and a date field.
  - Create a screen scenario.

  For more information, see Generate Screens for a Business Object [page 429].

Procedure

1. In the *Solution Explorer*, right-click your business object and select *Create Query*.
   The *Query Wizard* opens.
2. In the *Create a Query* step, under *Basic Information*, define your query by doing the following:
   1. Select a business object, if your solution has more than one business object.
   2. Select a parent node to link to the query, if your business object has more than one node.

      > → Tip
      >
      > The parent node is the starting point for the field selection in the *Select Query Fields* step. If a [1,n] association is defined for a field, you cannot use the field in the query result. Therefore, we recommend that you select the most detailed node as the parent node. For more information about associations, see Syntax of Business Object Definitions [page 191].
   3. Enter a query node name.
   4. Optional: Enter a description.

   By default, the query name is `QueryByElements`.
3. Optional: If you want to make your business object available in search queries, select *Include in Enterprise Search*; then do the following:
   1. Under *Enterprise Search Details*, define the following fields:
      - *Key Field*
        The field of the business object to be used in search queries to sort or filter search results by ID. You can either use the implicit alternative key field SAP_UIID or use an alternative key field created for the custom business object.
      - *Description Field*
        The field of the business object to be used in search queries to filter search results by description. The text fields of the anchor node of the query or the subnodes of the anchor node of the query can be used.
      - *Last Changed Field*
        The field of the business object to be used in search queries to sort or filter search results by last change date. The Date/Date Time fields of the custom business object can be used.

2. Under *UI Navigation*, define the screen you want to open when an end user selects this business object in the search results:

SAP Business ByDesign	Other SAP Cloud Solutions
In the *Navigation Target UI* field, select the quick activity floorplan (`.QA.uicomponent`) of your solution.	In the *Thing Type* field, select the thing type floorplan (`_TT`) of your solution.
When you select the *Operation*, the *Inport* field is filled automatically.	Additionally, you can also select *Enable Feed*, if you want to reference the business object in the *Feed* view.

3. Click *Next*.

4. Optional: If you want to enable read access logging for sensitive personal data, select *Log Sensitive Personal Data*.
5. In the *Select Query Fields* step, select the fields you want to include in the query.
6. In the *Define Query Parameters* step, refine the properties of each query parameter.
   1. Optional: Change the parameter field name.
   2. Optional: Select the *Selection* checkbox if you want to include the parameter as a search parameter.
   3. Optional: Select the *Basic Find* checkbox if you want the parameter to be used in a text search.
   4. Optional: Select the *Result* checkbox if you want the parameter to be displayed in the query result.

      > **i Note**
      >
      > You cannot use fields with [1,n] associations in a query result.

   5. Click *Next*.
7. In the *Review* step, check that the information you have entered is correct and then click *Finish*.
   A new item with the file extension `.qry` appears in the *Solution Explorer*.
8. In the *Solution Explorer*, right-click the business object and select *Activate*.

Result

You have created a custom query in your solution. In the *Solution Explorer*, a `[query name].qry file` ( ) is displayed. You can now use the scripting language to:

- Access your query
- Set the query parameters in order to be able to return a list of business object instances based on these parameters
- Execute the query
  For more information, see Execute a Query to Explore a Business Object [page 52].

For more information, see Scripting Language [page 189].

You can create an object work list (OWL) and use the UI designer to design the screen to use the business object query data. For more information, see Model an Object Work List [page 703] and Configure a Query [page 741].

If you want to use the query to make a custom business object available in the enterprise search, do the following: In the UI designer, assign the custom business object as an enterprise search object to the work

center view. For more information, see Model a Work Center View [page 710] under “Define the enterprise search objects”.

The custom business object is available in the enterprise search, once the target UI has been assigned to a user. For information about assigning access rights, see the “Business Users Quick Guide” in the documentation of the SAP cloud solution.

Related Information

Define Access Control [page 424]

8.3.3.5 Define Access Control

Overview

In the studio, you can implement instance-based access control for a business object you created by referencing an SAP business object for which an access context is defined.

Access rights in the cloud solution are based on work center and work center views. For each view, an access context is predefined that defines for which business data administrators can restrict user access rights, for example, by company, site, sales, or employee. For example, if business users have access rights for a work center view but should only see company-specific data, you can define your business object so that it inherits the access control list (ACL) from an SAP business object that is assigned the company access context.

Prerequisites

- An SAP access context exists that you can use for your use case.
  For more information about access contexts and access restrictions, see the relevant documentation of the cloud solution.
- An SAP business object exists with the necessary access context and an access control list (ACL) is directly assigned to this business object.

Procedure

1. In the studio, define a business object with an association that is marked as relevant for access control by doing the following:
   1. Create a business object, for example, with the name `AccessControlByCompany`, and enter your business object definition.
      For information about creating and defining a business object, see Business Objects Quick Guide [page 398].

2. Add an element to the business object root node that you use to find the relevant business object instance.

   Example

   You want the access context to be controlled by company, so you define a `Company` element of type `ID`:

   ```
   element Company : ID;
   ```

3. Add an association to the business object that is used for access control and mark the association with the annotation `[RelevantForAccessControl]`.

   Note

   - For each business object, you can only use one association that is relevant for access control.
   - The source node of the association must be the root node.
   - The target node of the association must be root node of the target business object, that is, the controlling business object.
   - The target business object must have an association that is relevant for access control.
   - The target of this association must be the `AccessControlList` dependent object.

   Example

   ```
   [RelevantForAccessControl] association For_Access_Control to
   CompanyFinancialsProcessControl;
   ```

4. Add a status element to the business object root node to handle errors, typically `ConsistencyStatusCode`:

   ```
   element Consistent :ConsistencyStatusCode;
   ```

2. In the studio, create a query using the *Query Wizard* to include the query data in an object worklist (OWL). For information about creating a query, see Create a Query [page 421].

   Note

   For correct instance handling, a query based on Fast Search Infrastructure (FSI) is necessary. By default, the query includes the ACL data even if you do not select this information.

3. Create an `Event-BeforeSave.absl` script file and define an association for access control.
   For information about creating script files, see Define the Business Logic for a Business Object [page 428].
   1. In the `Event-BeforeSave.absl` script file, add the query you created to get the instance for access control.

      Example

      ```
      Query = CompanyFinancialsProcessControl.QueryByCompany;
      Selection = QueryCreateselectionParams();
      Selection.Add(Query.CompanyID, "I", EQ", this.Company.Content);
      QueryResults = Query.Execute(Selection);
      ```

      In this example, access is controlled by company and the query uses the company at the root node as the basis for selection.

2. Find the relevant instance in the query and set the association to this instance.

   Example

   ```
   foreach (companycontrol in QueryResults) {
   this.For_Access_Control = companycontrol;
    break;
   }
   ```

3. For error handling, set a status on the business object root node.

   Example

   ```
   if (this.For_Access_control.IsSet ()){
   this.Consistent = "3";
   }
   else{
    this.Consistent = "2";
   }
   ```

   In this example, the consistency status codes are `"2"` for consistent and `"3"` for inconsistent.

4. Assign the access context to the work center view.
   1. In the studio, generate the screens for your business object.
      For information about generating screens, see Generate Screens for a Business Object [page 429].
   2. Open the work center view for editing by right-clicking the `*WCVIEW.uiwocview` screen file in the *Solution Explorer* and selecting *Open in UI Designer*.
      The view opens in the UI designer.
   3. In the *Properties* window, under *RBAMData*, select *Access Context Code* and assign the same access context code that is assigned to the associated SAP business object.
   4. Save and activate your changes.
   5. Check that the correct screens are assigned: Under *RBAM Data*, select *Assigned Objects*.
      The system displays the directly linked OWL and other linked screens that you can navigate to from the OWL, for example, an object instance floorplan (OIF) or a quick activity floorplan (QAF).
5. Assign the business object that controls access to the linked screens by doing the following:
   1. In the studio, double-click the screen file, for example, `*QA.uicomponent` or `*OIF.uicomponent`.
      The screen opens in the UI designer.
   2. In the *Properties* window, under *RBAMData*, check that the following entries exist:
      - *Access Check Mode* is set to *Privileged Except Access*.
      - *Authorization Classification Code* is set to *Application*.
   3. Open the collection of the *Access Controlled Business Object*.
      The *Select Business Objects* dialog box opens.
   4. Check that the correct business object is selected and that the *Unrestricted Access* checkbox is not selected.
   5. Save and activate your changes; then close the UI designer.

Result

You have defined an association for access control, defined the access context for the work center view, and defined access control for the screens for which you want to restrict access rights. You can now manage access rights for business users in the cloud solution. For information, see Assign Access Rights [page 427].

8.3.3.6 Assign Access Rights

Overview

If you have implemented instance-based access control for a business object you created in the studio, you can restrict read and write access in the SAP cloud solution on the basis of a work center view's access context.

In the SAP cloud solution, administrators can assign access rights to each employee who has a user in the system. For example, if an employee should only see data for a specific company and the company access context is assigned to a work center view, you can assign or restrict access to company-specific data for this view.

Prerequisites

You have defined associations that are relevant for access control in your business object and assigned the SAP business object that controls access to the work center views. For information, see Define Access Control [page 424].

Procedure

1. In the SAP cloud solution, disable the access rights of all users for the relevant work center views.
    1. Open the *Application and User Management* work center ▶ *User and Access Management* ▶ *Business User* subview.
    2. Select the user whose access rights you want to assign using the access control settings you have defined for your business object; then click *Edit* and choose the *Access Rights* option to open the access rights editor.
    3. On the *Work Center and View Assignment* tab, find your work center in the list of available work centers and clear the *Assigned to User* checkbox.
    4. Save your changes and log off from the system.
    Your changes are applied the next time you log on.
2. In the studio, update the authorization and access rights by doing the following:
    - In the *Solution Explorer*, right-click your solution and select *Update Authorization and Access Rights*.

> **Note**
>
> You must disable the access rights of all users before you update authorization and access rights in the studio.

3. Assign access rights to users for work centers views.
    1. In the SAP cloud solution, open the *Application and User Management* work center ▶ *User and Access Management* ▶ *Business User* subview.
    2. Select the user whose access rights you want to edit; then click *Edit* and choose the *Access Rights* option to open the access rights editor.
    3. On the *Work Center and View Assignment* tab, in the list of available work centers, find your work center and select the *Assigned to User* checkbox.
    4. Save your changes.
    If you do not want to restrict any access rights, you can log off from the system. Your changes are applied the next time you log on.
4. Restrict the access rights of users for work centers views.
    1. On the *Access Restrictions* tab, in the list of available work centers, find your work center and select the *Read and Write Access* checkbox.
    2. On the *Detailed Restrictions* tab, restrict read and write access on the basis of the work center views's access context, for example, based on the company.
    3. Save your changes and log off from the system.
    Your changes are applied the next time you log on to system.

8.3.3.7 Define the Business Logic for a Business Object

Overview

To define the business logic for your business object in the studio, you use event, action and validation script files.

For more information, see Actions, Events, and Validations [page 403].

Prerequisites

- You have saved the relevant business object.
- In order to define actions, you must have added them to your business object definition before. For more information, see the action [page 194] keyword in Syntax for Business Object Definitions [page 191].

Procedure

1. In the *Solution Explorer*, right-click your business object and select *Create Script Files*.

2. In the *Create Script Files* dialog box, for each of the business object nodes, select the events and validations that you want to implement.
   The system selects actions by default.
3. Optional: If you want script files to be mass-enabled, select the *Mass Enable* checkbox for the relevant actions, events or validations.
4. Click *OK*.
   The studio generates script files in the *Solution Explorer* with the file extension `.absl` . The script files appear below the corresponding nodes of your business object. For example, `Event-AfterModify.absl` for an event, `Validation-OnSave.absl` for a validation, or `Action-Release.absl` for an action. Each script file opens in a separate document window.
5. In the code editor, select the script file you want to edit and enter the code for the event, validation, or action.
   For more information about the methods and keywords you can use, see Syntax for Implementation of Business Logic [page 235].
6. Save your script files.

8.3.3.8 Generate Screens for a Business Object

Overview

In the studio, you can generate screens in the SAP cloud solution based on your business object definitions, for example, work centers, floorplans, and fact sheets.

In the studio, you can generate screens in the SAP cloud solution based on your business object definitions. For example, you can generate work centers, floorplans, and fact sheets. These screens then automatically contain an initial set of the fields that you have defined for your business object. The screens behave according to the business logic that you have implemented for your actions and events. You use the user interface designer to enhance the screens and adapt them to your needs.

> → Tip
>
> You can also use your generated screens for mobile solutions. For more information, see step 3 of this document and Mobile Solutions [page 774].

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Prerequisites

- If you generate screens based on an SAP business object, the business object must be released for the SAP public solution model (PSM).
- If you generate screens based on your business object, the business object must be activated. For more information, see Business Objects Quick Guide [page 398].

Procedure

1. In the *Solution Explorer*, do one of the following:
   - If you generate screens based on an SAP business object, proceed as follows:
     1. Right-click the project in your solution and select *Add* > *New Item*.
     2. Select the *Business Object* template, enter a name and click *Add*.
   - If you generate screens based on one of your business objects, right-click the business object and select *Create Screens*.
2. In the *Create Screens* dialog box, enter a name for the new screen or set of screens.
   The system uses this name together with the abbreviation of the screen type as a file name as follows: `[screen name]_OWL.OWL.uicomponent`.
3. Under *Devices Supported*, you can select the devices on which a user can display the screens. The options available are *Desktop Only*, *Desktop and Tablet*, and *Mobile Only*. The default option is *Desktop Only*.

   > **i Note**
   >
   > You can change your selection in the *Property Explorer* of the UI designer. For more information, see Mobile Solutions [page 774].

4. Select either the screen types you want to create or a whole screen scenario with navigation paths.
   You can select the following screen types:
   - Object Instance Floorplan (OIF)
   - Quick Activity Floorplan (QAF)
   - Fact Sheet (FS)
   - Object Work List (OWL)
   - Object Value Selector (OVS)
   - Work Center View (WOCVIEW)
   - Quick View (QV), not available for SAP Business ByDesign
   - Quick Create (QC), not available for SAP Business ByDesign
   - Thing Inspector (TI), not available for SAP Business ByDesign
   - Thing Type (TT), not available for SAP Business ByDesign

   For detailed information on the various screen types, see Screen Types [page 408].
   Screen Scenario with Thing-based Navigation (not available for SAP Business ByDesign)
   If you select *Screen Scenario with Thing-based Navigation (not available for SAP Business ByDesign)*, the system generates the following screens and the navigation paths that you need to be able to navigate from one screen to the next:
   Object Work List, Quick Create, Quick View, Thing Inspector, Thing Type, Work Center, and Work Center View.
   Screen Scenario with Object-based Navigation

If you select *Screen Scenario with Object-based Navigation (called "Screen Scenario with Navigation" for SAP Business ByDesign)*, the system generates the following screens and the navigation paths that you need to be able to navigate from one screen to the next:
Object Instance Floorplan, Object Work List, Quick Activity Floorplan, Work Center, and Work Center View.

> **i** Note
>
> **Short ID**
>
> If you select *Work Center View* or a screen scenario containing a work center view, an additional field appears where you must enter a *Short ID* of up to 13 characters. It is preceded by the technical name of the solution and followed by **_WCVIEW**: `[Technical name of solution]_[Short ID]_WCVIEW`. The short ID helps you identify the work center view in the UI designer and in the SAP cloud solution. After you have entered the short ID, it is displayed in the following places:
>
> - UI designer: in the *Property Explorer* under *Component Info* in the *Short ID* field
> - SAP cloud solution: *Application and User Management* > *Work Center and View Assignment*

5. Click *OK*.

Result

In the *Solution Explorer*, the files for the screens are listed. The system also creates a port type package file (.ptp file) that contains the navigation, for example, from the object work list to the quick activity floorplan.

To view a screen, right-click a screen file and select *Preview Screen*. To edit a screen, right-click a screen file and select *Open in UI Designer*. For more information, see Introduction to the User Interface Designer [page 662].

8.3.3.9 Create an OWL Excel Export

Overview

Object Work Lists (OWL) generated for custom business objects allow you to export the OWL data to a Microsoft Excel file. The file is based on a standard template, which you cannot change. However, you can format the generated spreadsheet as required.

Prerequisites

- You have opened your solution containing an active custom business object.
- You have created a floorplan scenario with navigation.

Procedure

1. Log on to the SAP cloud solution and select *Application and User Management* > *Business Users*.
2. Select you user and select *Edit* > *Access Rights*.
3. In the *Work Center and View Assignment* view, select the work center you have created and assign it to your user. Click *Save and Close*.
4. Log off from the SAP cloud solution. Log on again and navigate to the work center you have assigned to your user.
   The work center you have created is displayed. It contains an *Export* button.
5. Click *New* and create several instances of your business object. .
6. Complete the fields and select *Save* and then *Close* to get to the entry screen again.
7. On the OWL, click *Go* to view the data records you have created.
8. Click *Export* > *To Microsoft Excel*.

Result

Microsoft Excel opens. The data records are displayed on a spreadsheet. You can save the data as it is or format it to suit your requirements.

Related Information

Example: Export OWL Data to Excel [page 432]

8.3.3.10 Example: Export OWL Data to Excel

Overview

In this example, you create a solution with a custom business object. You generate a screen scenario with object-based navigation and test the work center in the SAP cloud solution. After that, you create data records, which you export to Microsoft Excel.

Procedure

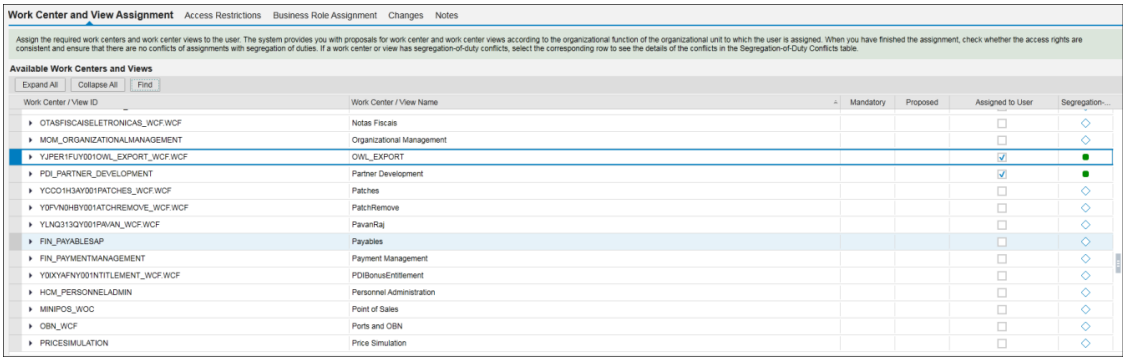
1. **Create a solution and generate the floorplan scenario.**
   1. Log on to the studio and create a new solution.
   2. In the *Solution Explorer*, click the *Add New Item* button.
      The *Add New Item* dialog opens.

3. Select *Business Object*, enter a name (for example, **`OWL_EXPORT.bo`**) and click *Add*.
   The *Business Object Selection* dialog opens.
4. Enter the following code:

```
import AP.Common.GDT;
businessobject OWL_EXPORT {
[Label("Identifier")] element ID:ID;
[Label("Name")] element  name:LANGUAGEINDEPENDENT_MEDIUM_Name;
[Label("Description")] element
description:LANGUAGEINDEPENDENT_MEDIUM_Description;
[Label("Start Date")] element  start_date:Date;
[Label("End Date")] element  end_date:Date;
[Label("Amount")] element  amount:Amount;
[Label("Title")] element  title:AcademicTitleCode;
```

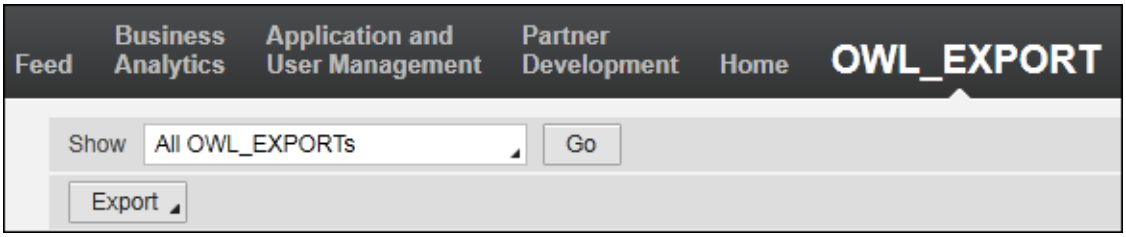
   }
   Save and activate the business object.
5. Right-click your solution and select *Create Screens*.
   The *Create Screens* dialog opens.
6. Select a screen scenario, enter a *Short ID*, and click *OK*.
   The screens are generated and the corresponding files are displayed in the *Solution Explorer*.

2. **Assign the work center and perform a test.**
   1. Log on to the SAP cloud solution and select *Application and User Management* > *Business Users*.
   2. Select you user and select *Edit* > *Access Rights*.
   3. In the *Work Center and View Assignment* view, select the work center you have created and assign it to your user. Click *Save and Close*.



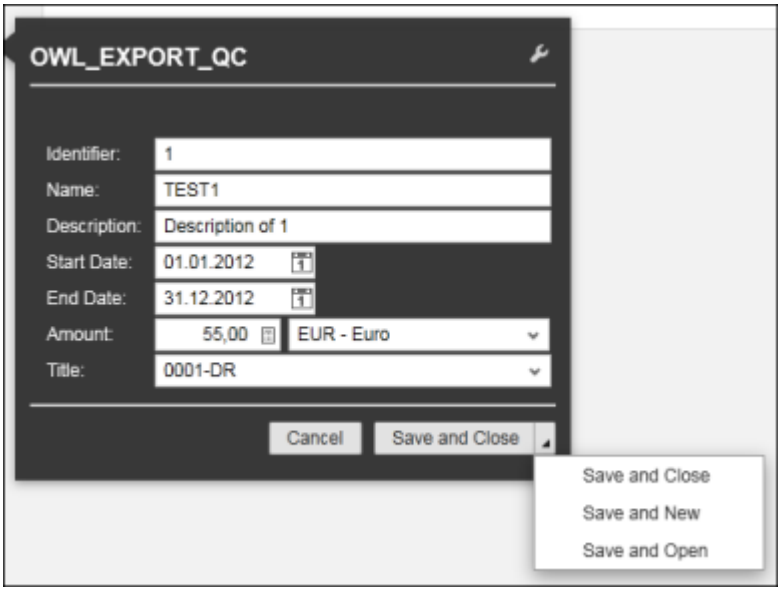
Work Center Assignment

4. Log off from the SAP cloud solution. Log on again and navigate to the work center you have assigned to your user.
   The work center is displayed. It contains an *Export* button.



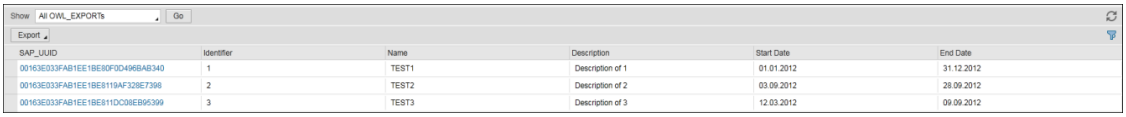
Export Button

5. Click *New*.
6. Create several instances of your business object. Complete the fields and select *Save* and then *Close* to get to the entry screen again.



Entry Screen

7. On the OWL, click *Go* to view the data records you have created.

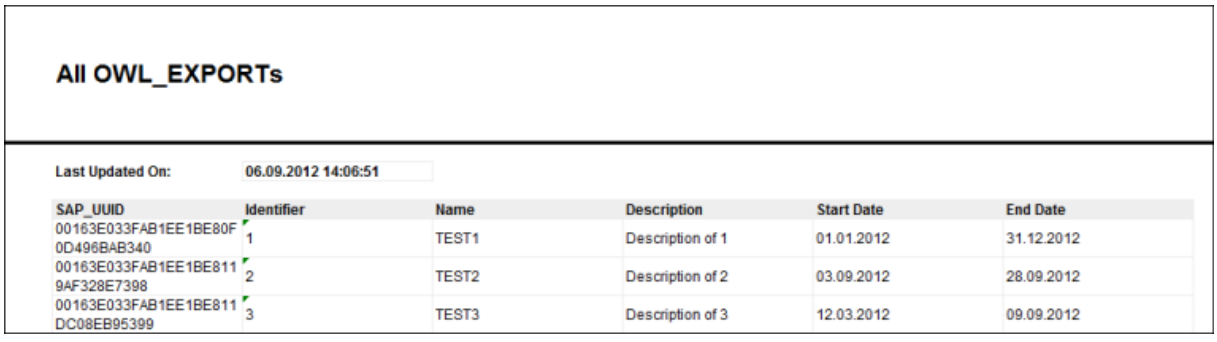


Data Records

8. Click *Export* > *To Microsoft Excel*.

Result

Microsoft Excel opens. The data records are displayed on a spreadsheet. You can save the data as it is or format it to suit your requirements.



Download

8.3.3.11 Enable Change History

Overview

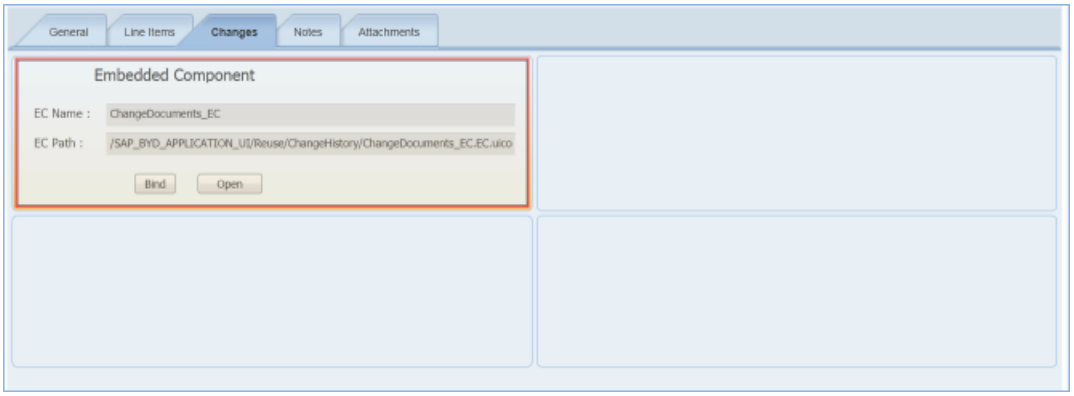
You can use an embedded component to ensure that a change history is written for all changes made to your custom business object. After adding the embedded component to your business object, you can display the change history in the SAP cloud solution on the *Changes* tab of your work center. All changes such as add, create, and delete are displayed in a list.

Prerequisites

- You have created users with the Developer and Business User roles. For more information, see User Setup Quick Guide [page 33].
- You have created and opened a solution in the studio. For more information, see Create a Customer-Specific Solution [page 177].
- You have created a business object or business object extension in your solution and entered a business object definition. At least one element/node/association in your business object is annotated with **ChangeHistory**. For more information, see Business Objects Quick Guide [page 398].
- You have created an object instance floorplan (OIF) for your business object. For more information, see Generate Screens for a Business Object [page 429].

Procedure

1. **Add an embedded component to the object instance floorplan (OIF) you created for your business object.**
    1. In the studio, in the *Solution Explorer*, double-click the object instance floorplan (OIF).
    The UI designer opens.
    2. On the *Designer* tab, click the *Changes* tab.
    3. In the *Configuration Explorer*, navigate to the `ChangeDocuments_EC` reuse component in the `SAP_BYD_APPLICATION_UI/Reuse/ChangeHistory` folder.
    4. Drag the reuse component to the top left pane of the OIF as shown in the graphic and then save your changes.



2. **Create an outport using the ECORootNodeID parameter.**
    1. On the *Controller* tab, right-click *Outports* and select *Add Outport*.
    The *Outport Configuration: OutPort* pane opens.
    2. Click the *Add Parameter* button.
    3. In the *ParameterName* field, enter `ECORootNodeID` and save your changes.
3. **Create a binding for the embedded component by binding the outport parameter and the ChangeHistory inport parameters.**
    1. Next to *ParameterBinding*, click ....
    The *Parameter Binding* dialog box opens.
    2. Select `NodeID` and click *OK*.
    3. On the *Changes* tab, click *Bind*.
    The *Embedded Component Binding* dialog box opens.
    4. Right-click *Navigations* and select *Add Navigation*.
    The *Navigation Details* are displayed.
    5. In the *OutPort* field, select `OutPort` and in the *InPort* field, select `InECONodeID`.
    6. Under *OutPort Parameters* and *InPort Parameters*, select the parameters and then click *Bind*.
    The outport and inport parameters are displayed under *Bound Parameters*.
    7. Click *OK* and save your changes.
4. **Create an event handler to fire the outport.**
    1. On the *Controller* tab, right-click *Event Handlers* and select *Add EventHandler*.
    The new event handler is displayed under *Event Handlers*.
    2. Right-click the new event handler and select *Rename*; then enter a new name for the event handler.
    3. Under *EventHandler Operations*, click the *Add Operation* button.
    4. Under *Type*, click in the field and then select *FireOutport*.
    5. Under *Configure the operation*, select your outport and save your changes.
5. **To ensure that the user can open the embedded component by clicking the Changes tab, associate the new event handler with the *OnClick* event.**
    1. Click the *Properties* tab.
    2. On the *Designer* tab, click the *Changes* tab and then click the *Properties* tab again.
    3. On the *Properties* tab, under *Events* > *OnClick*, select the event handler you created and then save your changes.
6. **Specify the configuration parameters in the `ChangeDocuments` embedded component properties.**
    1. Select the embedded component and click the *Properties* tab.

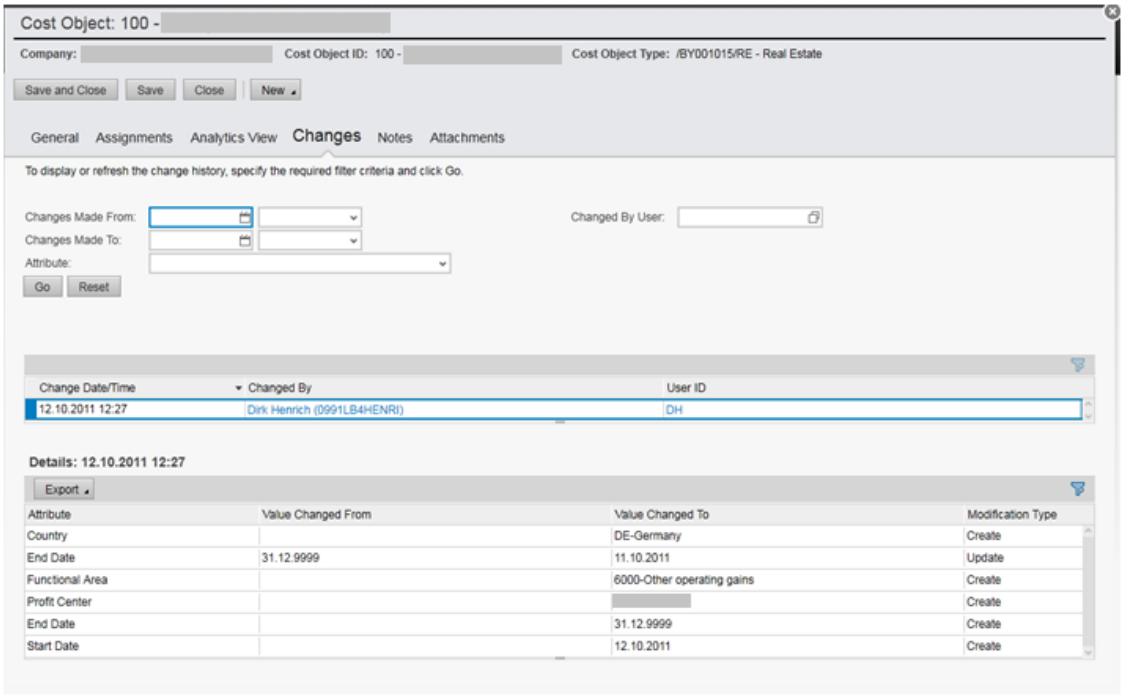
2. Under *Parameters*, specify the following parameters by clicking the arrow to the right of each field:

`BOName`	<Name of your business object>
`BONameSpace`	<Namespace of your business object>
`ECOName`	<Name of your business object>
`ECONameSpace`	<Namespace of your business object>

> **i Note**
>
> You can copy the name and namespace from the *BO Browser/Data Model* but copy and paste does not work. You can also copy and paste the namespace from the *Solution Properties* in the studio.

3. Deselect the *IsAttributeSearchEnabled* option.
4. Save and activate the floorplan.

7. **Create a business configuration set in the studio.**
   1. Activate your business configuration, right-click the solution and select *Deploy Business Configuration*. The system deploys the business configuration content you created and makes the field values available in the development environment.
8. **Test the embedded component.**
   1. Log on to the cloud solution as a business user and assign the work center that has the same name as your business object to your service agent.
   2. Navigate to the work center you created and click *Edit*.
   3. Enter data in one or more of the fields and save your changes.
   4. On the right of the screen, click *View All*.
   5. Select *Close* and *Refresh*.
      Your changes are displayed on the *Changes* tab.



Example of a change history

> **i** Note
>
> The **ChangeHistory** annotation does not work for Dependent Objects.

Related Information

Create a BC Set Using an SAP BCO [page 383]

8.4 Extensibility

8.4.1 Business Object Extensions

8.4.1.1 Business Object Extensions Quick Guide

Business object extensions enable you to enhance business objects that are provided by SAP by adding extension fields to them.

After you create and activate a business object extension, you can add the extension field to the corresponding forms, screens, reports, and search categories.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Business and Technical Background

Business Object Extensions

You can customize the business objects that are provided by SAP by creating business object extensions. When you create a business object extension, you modify the extendable nodes of an existing business object to include additional fields. Once the business object extension has been activated, you can include the extension fields in script files, screens, reports, and search categories. Creating a business object extension does not create a copy of the business object. Once you extend a business object, the original business object is no longer available and is replaced by the extended business object.

Most business objects provided by SAP contain both extendable and non-extendable nodes. However, some business objects do not contain any extendable nodes and cannot be used with business object extensions. You can only create business object extensions for business objects that are provided by SAP. Other business objects cannot be used with business object extensions.

Tasks

Create a Business Object Extension

You can create an extension for an SAP business object (business object extension) and add new fields to the extendable nodes of this business object. The extendable nodes are automatically provided in a template, to which you can add additional fields.

For more information about this task, see here [page 442].

Define the Business Logic for a Business Object Extension

You use event script files and validation script files to define the business logic for your business object extension. For more information about events and validations, see Actions, Events, and Validations [page 403].

For more information about this task, see here [page 444].

Add an Extension Field to a Screen

You can add an extension field to a business object's corresponding screens.

For more information about this task, see here [page 445].

Add an Extension Field to a Form

You can add an extension field to the business object's corresponding forms. You can either add the extension field to an original form, or you can create a copy of a form and add the extension field to the copy.

For more information about this task, see here [page 446].

Add an Extension Field to a Report

You can add an extension field to a data source that is based on the same business object as the extension field. You can then add the field to any reports based on the data source.

For more information about this task, see here [page 448].

Add an Extension Field to the Enterprise Search

You can add an extension field to an enterprise search category. This allows users to search for the contents of an extension field within a particular category, such as sales orders or suppliers.

For more information about this task, see here [page 450].

Define a Message for a Business Object Extension

You can define a message in a business object extension and raise the message in an event. Messages can only be raised in the business object extension where they are defined.

For more information about this task, see here [page 451].

Add a Label and Tooltip to an Extension Field

You can define a label and a tooltip for an extension field. If you then add the extension field to a screen:

- the label is displayed on the screen instead of the extension field name

- the tooltip is displayed when you place the cursor on the extension field

For more information about this task, see here [page 449].

Related Information

Process Extension Scenario Quick Guide [page 459]
Syntax for Business Object Extension Definitions [page 218]
Maintenance of Business Object Extensions [page 441]

8.4.1.2 Maintenance of Business Object Extensions

Overview

A customer-specific solution is in maintenance mode, if it has been assembled and downloaded; the solution status then is *Assembled*. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.

Change and Delete Restrictions

The following table gives an overview of the change and delete restrictions for business object extensions in a solution that is in maintenance mode. The system performs the change and delete restrictions checks when you save the business object extension.

Topic	Adding Allowed	Deletion Allowed	Renaming Allowed	Comments
Business object extension	Yes	No	No	
Business object node element	Yes	No	No	
Data type of element	No	No	No	
`Label` annotation	Yes	Yes	Yes	
`Tooltip` annotation	Yes	Yes	Yes	
`Reference` annotation	Yes	No	No	

Topic	Adding Allowed	Deletion Allowed	Renaming Allowed	Comments
`Relation` annotation	Yes	No	—	The target must not be changed.
`Decimal` annotation	No	No	No	
`Scenario` annotation	Yes	No	No	
Messages	Yes	Yes	Yes	
Process extension scenario	Yes	No	No	
Providing default values	Yes	Yes	—	Changes are allowed.

Related Information

Business Object Extensions Quick Guide [page 438]

8.4.1.3 Tasks

8.4.1.3.1 Create a Business Object Extension

Overview

You can create an extension for an SAP business object (business object extension) and add new fields to the extendable nodes of this business object. The extendable nodes are automatically provided in a template, to which you can add additional fields.

Prerequisites

You know how to work with business objects and how to use the scripting language.

Procedure

1. In the *Solution Explorer*, expand your solution, right-click your project file, and select *Add* > *New Item*.

The *Add New Item* dialog appears.

2. Select *Business Object Extension*.
3. Enter a *Name* for your business object extension and click *Add*.

> **i Note**
>
> You cannot use a name that is already used for a business object or business object extension in your solution.

The *Business Object Selection* dialog appears.

4. Select the *Namespace* of the SAP business object that you want to extend.
5. Select the *Business Object* that you want to extend.
6. Click *OK*.
   A business object extension item with the file extension `.xbo` is added to your project. This file contains a template for the business object extension, including the extendable nodes.
7. In the *Solution Explorer*, double-click the business object extension to open the file in the editing pane.
8. To add extension fields, add elements to the extendable nodes with the scripting language.
9. Click the *Save* button.
10. Right-click the business object extension and select *Activate*.

Example

This example shows an extension of the `Lead` business object provided by SAP. Two text elements, `Test_Text1` and `Test_Text2` , have been added to the business object extension – one at the root level and one at the item level.

```
import AP.Common.GDT;
import AP.CRM.Global;
[Extension] businessobject AP.CRM.Global:Lead {
  element Test_Text1:LANGUAGEINDEPENDENT_EXTENDED_Text;
  node Item {
    element Test_Text2:LANGUAGEINDEPENDENT_EXTENDED_Text;
  }
}
```

Related Information

Syntax for Business Object Extension Definitions [page 218]
Business Object Extensions Quick Guide [page 438]
Scripting Language [page 189]
Business Objects Quick Guide [page 398]

8.4.1.3.2 Define the Business Logic for a Business Object Extension

Overview

You use event script files and validation script files to define the business logic for your business object extension. For more information about events and validations, see Actions, Events, and Validations [page 403].

Prerequisites

The extended business object must be activated if you want to access an extension field in your script file.

Procedure

1. In the *Solution Explorer*, right-click your business object extension and select *Create Script Files*.
   The *Create Script Files* dialog box is displayed, listing all the nodes of the business object. You can only select nodes that are released as part of the public solution model (PSM).
   To find out which nodes have been released for the PSM, select the `Repository Explorer` in the studio. For more information, see Repository Explorer [page 49]
2. Select the script files that you want to create for each of the nodes of the business object.
   Events, such as `BeforeSave` and `AfterModify`, can be created for each node. The validation `OnSave` can only be created for the root node.
3. Select the *Mass Enable* checkbox if you want the script file to be mass enabled.
   If mass enabled, the script file will be executed for multiple instances of the business object node. You cannot clear this checkbox after a script file has been created. If you want to modify a script file afterwards so that it will only be executed for a single instance of the business object node, you need to delete the script file first and re-create it again.

   > **i Note**
   >
   > If you use the `this` keyword in the coding of a mass enabled script file, `this` represents a collection of business object node references.

4. Click *OK*.
   In the *Solution Explorer*, script files with the file extension `.absl` appear below the corresponding nodes of your business object. Each script file is opened on a tab page in the code editor.
5. In the code editor, select the script file you want to edit and enter the coding.
   For more information about the methods and keywords you can use, see Syntax for Implementation of Business Logic [page 235].
6. In the *File* menu, click *Save*.
7. In the *Solution Explorer*, right-click the script file and select *Activate*.

8.4.1.3.3 Add an Extension Field to a Screen

You can add an extension field to a business object's corresponding screens.

> **i** Note
>
> This function is not available for solution templates. For more information, see Solution Templates [page 174].

Prerequisite

You have created and activated a business object extension that contains at least one extension field.

Procedure

1. In the *Solution Explorer*, right-click the business object extension (`.xbo` file) and select *Enhance Screen*.
   The *Enhance Screen* dialog appears and lists the screens that can be modified for this business object extension.
2. Select the screen to add the extension field to, and click *OK*.
   The UI designer opens.
3. In the *Extensibility Explorer*, select the *Section Group* or *List* to which you want to add the extension field.
4. Depending on the type of entity selected, click either the *Add Extension Field to Section Group* button or the *Add Extension Field to List* button.
   The *Add Extension Field* dialog appears and lists the available extension fields.
5. Select the extension fields to add to the screen by clicking the *Add* check box.

   > **i** Note
   >
   > If you want the field to be read-only, select the *Display Only* check box.

6. Click *Apply*.
7. Click the *Save* button.
8. Click the *Activate* button.
   The *Activate Worklist* dialog appears.
9. Select the screen you have modified and click *OK*.

Related Information

Create a Business Object Extension [page 442]

8.4.1.3.4 Add an Extension Field to a Form

You can add an extension field to the business object's corresponding forms. You can either add the extension field to an original form, or you can create a copy of a form and add the extension field to the copy.

> **i** Note
>
> This function is not available for solution templates. For more information, see Solution Templates [page 174].

Feature Matrix

> **⚠** Caution
>
> This documentation covers the SAP Cloud Applications Studio ("studio") as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Prerequisites

- You have created a solution.
- You have created and activated a business object extension in that solution.

Procedure

1. **Switch on admin mode**
   In the studio, right-click your solution and, in the *Administration* menu, select *Switch On/Off Admin Mode*.
2. **Create a Copy of a Form (optional)**
   You can create a copy of a form and add the extension fields to the copy.

   > **i** Note
   >
   > This option is useful if you want to be able to print different versions of a form. Example: Depending on the selection by the business user, the preview is to show the standard print form or the extended print form.

   1. Right-click your business object extension (`.xbo` file) and click *Enhance Form*.
      The *Enhance Form* dialog appears.
   2. Click *Copy Form*.

The *Form Template Maintenance* screen appears.

> **i Note**
>
> If you are not already logged on to the SAP solution, you will be prompted to log on.

3. In the *Show* list, select *Active Forms* and click *Go*.

   > **→ Tip**
   >
   > To speed up the search, click *Advanced*. Enter the template name or template group and click *Go*.

   The available forms are displayed.
4. Select the form template that you want to copy, and click *Copy* > *As New Template*.
5. Enter a *Template Name* and *Template Description* for your new template.
6. Click *OK*.
7. Click the *Refresh* button to verify that your template has been added to the list of available templates.

   > **→ Tip**
   >
   > Use a filter to find your template by name.

3. **Edit the Form in the Easy Form Editor**
   1. Right-click your business object extension (`.xbo` file) and click *Enhance Form*.
      The *Enhance Form* dialog appears.
   2. Select the extension field you want to add to the form and click *OK*.
      The *Further Usage of Extension Field* screen appears.

      > **i Note**
      >
      > If you are not already logged on to the SAP solution, you will be prompted to log on.

   3. Select the form template to which you would like to add the extension field and click *Add Field and Edit* > *Easy Form Editor*.
      The *Easy Form Editor* screen appears.
   4. In the *Sections* area, select the section to which you would like to add an extension field.
      The *Details* area lists the fields that can be added to the section.
   5. In the *Details* area, select the extension field to add to the form, and set the *Show* attribute to *Show always*.
   6. To preview the form with the extension field, in the *Preview* area, click the *Refresh* button.
   7. To save your changes, click *Save*.
   8. To make your change visible to other users, click *Publish and Close*.

Additional Activities

To view the form template in the SAP solution, business users must create a form template rule.

For more information, see “Create a Form Template Rule“ in the “Form Template Selection Quick Guide" in the documentation of the SAP cloud solution.

Related Information

Create a Business Object Extension [page 442]

8.4.1.3.5 Add an Extension Field to a Report

You can add an extension field to a data source that is based on the same business object as the extension field. You can then add the field to any reports based on the data source.

> **i** Note
>
> This function is not available for solution templates. For more information, see Solution Templates [page 174].

The following table shows whether an extension field can be used as a key figure or characteristic.

Extension Field Type	Analytics Usage
Amount	Key Figure
Date	Characteristic
Decimal Number	Key Figure
E-Mail Address	Characteristic
Indicator	Characteristic
List	Characteristic
Text	Characteristic
Time	Characteristic
Quantity	Key Figure
Web Address	Characteristic

Prerequisite

The business object extension has been activated.

Procedure

1. Open your solution and select *Administration* > *Switch On/Off Admin Mode in SAP Solution*.
2. Right-click on your business object extension (`.xbo` file) and click *Enhance Report*.
   The *Enhance Report* dialog box appears.
3. Select the extension field you want to add to the report and click *OK*.
   The *Further Usage of Extension Field* screen appears.

   > **i** Note
   >
   > If you are not already logged on to the SAP cloud solution, you will be prompted to log on.

4. On the *Data Sources and Reports* tab, you can view all data sources available for the business context in which you created the field. When you select a data source, all reports based on that data source are displayed in the lower table.
5. To add the field to a data source, select the data source and click *Add Field*.
6. To add the field to an existing report based on this data source, select the report from the lower table and click *Add Field*.
   The report opens in admin mode.
7. In the *Add Fields* dialog box, select the *Display in Report* checkbox next to the extension field.
   Characteristics are added to the report under *Not Currently Shown* and key figures are available under *Columns*.
8. Click *OK* and close the report without saving.
   The field will be available to all business users when they next access the report.

Related Information

Create a Business Object Extension [page 442]

8.4.1.3.6 Add a Label and Tooltip to an Extension Field

Overview

You can define a label and a tooltip for an extension field. If you then add the extension field to a screen:

- the label is displayed on the screen instead of the extension field name
- the tooltip is displayed when you place the cursor on the extension field

Procedure

1. Create an extension field with a label and a tooltip using the following syntax:

   ```
   [Label ("<label name>")] [Tooltip ("<tooltip name>")] element <extension
   field name> : <data type>;
   ```

   For more information, see Create a Business Object Extension [page 442].
2. Add the extension field to a screen. For more information, see here [page 445].

Result

In the *Solution Explorer* in the studio, right-click your extended screen (`.xuicomponent`) and select *Preview*.

The label is displayed for the extension field. The tooltip is displayed when you place the cursor on the field.

8.4.1.3.7 Add an Extension Field to the Enterprise Search

You can add an extension field to an enterprise search category. This allows users to search for the contents of an extension field within a particular category, such as sales orders or suppliers.

> **i Note**
>
> This function is not available for solution templates. For more information, see Solution Templates [page 174].

The following table shows which type of search can be used for each extension field type:

Search Type by Extension Field Type

Field Type	Basic Worklist Search	Advanced Worklist Search	Basic Enterprise Search	Advanced Enterprise Search
Amount	No	Yes	No	Yes
Date	No	Yes	No	Yes
Decimal Number	No	Yes	No	Yes
E-Mail Address	Yes	Yes	Yes	Yes
Indicator	No	Yes	No	Yes
List	Yes	Yes	Yes	Yes
Text	Yes	Yes	Yes	Yes

Field Type	Basic Worklist Search	Advanced Worklist Search	Basic Enterprise Search	Advanced Enterprise Search
Time	No	No	No	No
Quantity	No	Yes	No	Yes
Web Address	Yes	Yes	Yes	Yes

Prerequisite

The business object extension has been activated.

Procedure

1. Right-click your solution and select *Administration* > *Switch On/Off Admin Mode in SAP Solution*.
2. Right-click your business object extension (`.xbo` file) and click *Enhance Enterprise Search*.
   The *Enhance Enterprise Search* dialog box appears.
3. Select the extension field you want to add to the search and click *OK*.
   The *Further Usage of Extension Field* screen appears.

   > **i** Note
   >
   > If you are not already logged on to the SAP cloud solution, you will be prompted to log on.

4. On the *Enterprise Search* tab, you can view all enterprise search categories to which you can add the field. This is determined by the business context in which you created the field.
5. To add a field to an enterprise search category, select it from the list and click *Add Field*.
   The enterprise search category determines in which category users will be able to search for the contents of this field using the basic enterprise search. They can then refine their search by clicking on *Advanced* and using the extension field as a search parameter.

8.4.1.3.8 Define a Message for a Business Object Extension

Overview

You can define a message in a business object extension and raise the message in an event. Messages can only be raised in the business object extension where they are defined.

The following message types are available:

- Information

- Warning
- Error
- Success

Prerequisites

- You have an open solution in the studio.
- You have created a business object extension (`.xbo`) as part of your solution. This extension contains at least one element and is activated.

Procedure

1. In the *Solution Explorer*, double-click the business object extension to open the file in the editing pane.
2. Use the `message` and `text` keywords to specify the name and content of the message, respectively.
3. Use the `raises` keyword to specify which messages are raised for the individual nodes of the business object extension.

   ### Example

   ```
   import AP.Common.GDT;
   import AP.CRM.Global;
   [Extension] businessobject AP.CRM.Global:Opportunity raises
   Information_Message, Error_Message, Warning_Message {
   // You must activate this business object before you can access the
   extension fields
   // in script files and screens.
   message Information_Message text "This is an Information message";
   message Error_Message text "This is an Error message with one parameter
   &1" : LANGUAGEINDEPENDENT_SHORT_Name;
   message Warning_Message text "This is a Warning message with two
   parameters &1 and &2" : LANGUAGEINDEPENDENT_SHORT_Name, IntegerValue;
   message Success_Message text "This is a Success message with a variable
   parameter &1" : Amount;
   element Test_Extension_Field : Amount;
   node Item raises Information_Message, Success_Message {
   }
   }
   ```

   ### Note

   You can include up to four parameters. For each parameter, you must specify the data type.

4. In your action or event script file, raise the message using the `raise` keyword.

   ### Example

   ```
   import ABSL;
   raise Information_Message.Create("I");
   raise Error_Message.Create("E", "Error");
   raise Warning_Message.Create("W", "Warning", 25);
   raise Success_Message.Create("S", this.Test_Extension_Field);
   ```

> **Note**
>
> Code completion is available after you type the `raise` keyword.

Result

Messages appear on the user interface based on the actions or events that you defined.

Related Information

Scripting Language [page 189]
Business Object Extensions Quick Guide [page 438]

8.4.1.3.9 Create References to Customer-Specific Fields

Customers can enhance their SAP cloud solution by creating extension fields in the adaptation mode. In the studio, you can create references to these customer-specific fields. After that, you can implement the business logic for the referenced fields in script files of your solution.

> **Note**
>
> - You cannot reference a customer-specific field in a solution template.
> - You cannot reference a calculated field.
> - You can only reference a customer-specific field in the extension of a business object that has been released in the public solution model (PSM).

What you need to know about the lifecycle management of field references

Uploading Field References

If you upload your solution with references to customer-specific fields, these fields must also exist in the SAP cloud solution. If this is not the case, you need to create these fields manually in the cloud solution. The technical name and the type of the fields must always be identical in your solution and in the SAP cloud solution.

Deleting Field References

As long as a customer-specific field is referenced by your solution, it cannot be deleted anymore. If you want to delete a reference to a customer-specific field in your solution, make sure that you delete it in the script file in the studio first.

Prerequisites

- In the SAP cloud solution, your customer has created extension fields for a business context.
- In the studio, you have created a solution containing a business object extension and at least one script file. In the business object extension, make sure that you have extended the business object corresponding to the business context of the extension field.

Procedure

1. In the *Solution Explorer*, right-click your solution and select *Add New Item*.
   The *Add New Item* dialog opens.
2. Select *SAP* > *Extension* > *References to Customer-Specific Fields*, enter a name and click *Add*.
   A `.ref` file ( ) appears in the *Solution Explorer*.
3. A document window opens that displays the *References to Customer-Specific Fields* table. In the table, select the fields that you want to use in your solution by clicking the *Used* field. You can sort the table by clicking the column titles.

   > → Tip
   >
   > The table shows the fields that are available for each business object or business object node. If a field is used in several business contexts in the SAP cloud solution, the field is displayed several times in the table with the same field name. The *Business Object* column helps you identify the correct field.

4. Save and activate the `.ref` file.

   > ℹ Note
   >
   > You can only create one `.ref` file per solution. It contains the references to all customer-specific fields. If you want to add further references, open the `.ref` file and select additional fields you want to reference.

Result

You can now define the business logic for the referenced fields in the relevant script files of your solution. The referenced fields are shown in the code completion.

You can delete references to customer-specific fields when your solution is in maintenance mode. For more information, see Maintenance Mode [page 185].

Related Information

Example: Calculate the Duration of a Campaign (References to Customer-Specific Fields) [page 455]

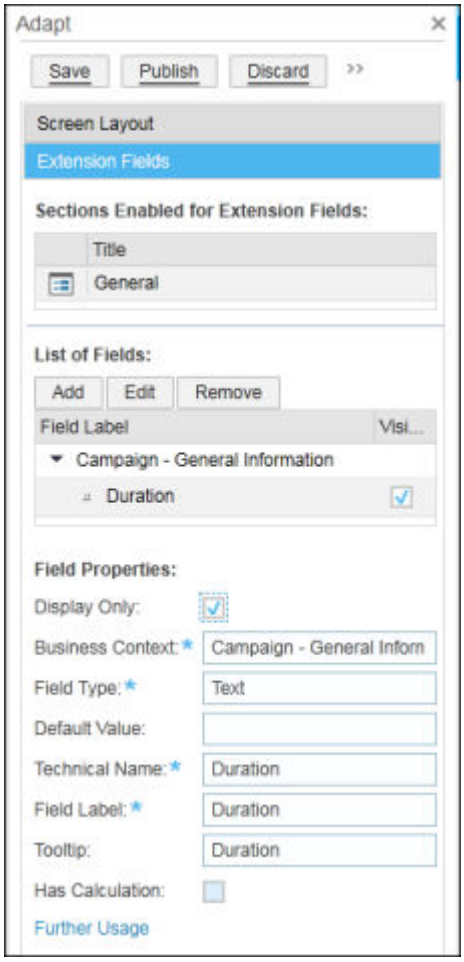
8.4.1.3.10 Example: Calculate the Duration of a Campaign (References to Customer-Specific Fields)

Example

In SAP Cloud for Customer, your customer has created a **Duration** field in the **Campaigns** work center view. In the studio, you create a reference to that field and use it in a script file to calculate the duration of a campaign.

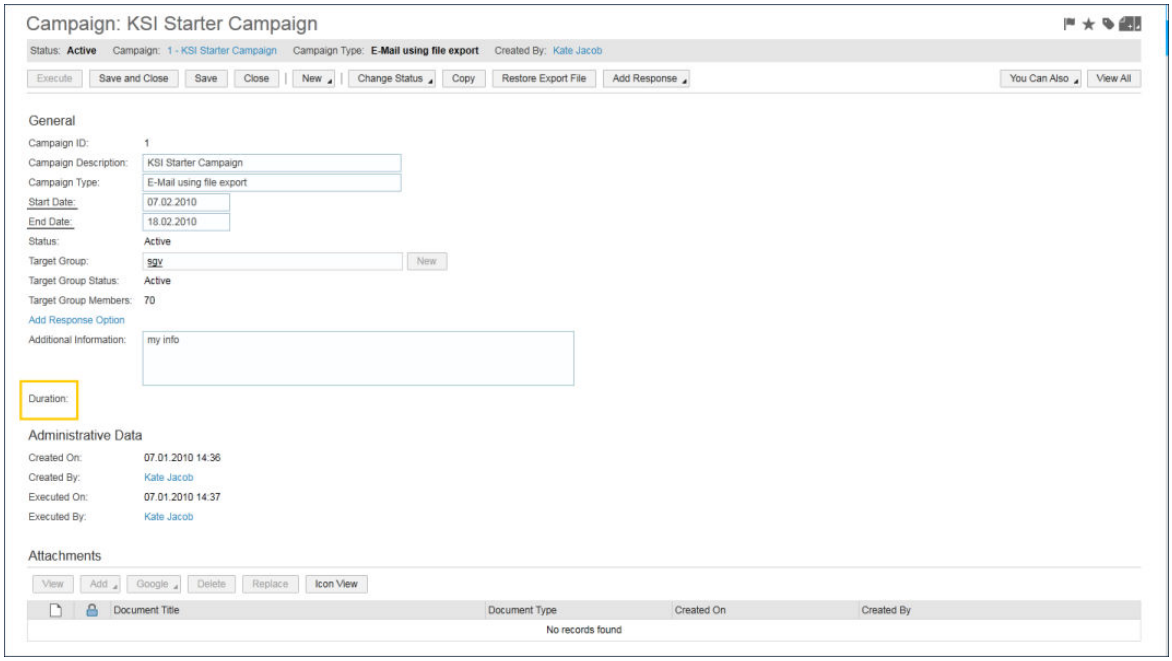
Create an extension field in the cloud solution.

1. Log on to the cloud solution with a business user and choose *Marketing* > *Campaign Management* > *Campaigns*.
2. Select a campaign and click *Edit*.
3. Choose *Adapt* > *Enter Adaptation Mode* and then *Adapt* > *Edit Screen*.
4. In the *Adapt* dialog, choose *Extension Fields* > *Add*.
5. In the *Add New Field* dialog, in the *Field Label* field, enter `Duration` and click *Save*.
6. In the *Adapt* dialog, select the *Visible* and *Display Only* fields.



Adapt Dialog

7. Click *Save* and *Publish* and confirm the *Publish Changes for Screen* dialog.
   The adaptation mode closes and you can see the field you have added on the screen.



Duration Field on the Adapted Screen

8. Log off from the cloud solution.

Create a reference to the customer-specific field in the studio.

1. Log on to the studio and create a new solution.
2. In the *Solution Explorer*, click *Add New Item*.
   The *Add New Item* dialog opens
3. Select *SAP* > *Extension* > *Business Object Extension*.
4. Enter a name (**`Campaign.xbo`**) and click *Add*.
5. Right-click the `.xbo` file and click *Save and Activate* ( ).
6. In the *Business Object Selection* dialog, select a namespace (`http://sap.com/xi/AP/CRM/Global`) and a business object (`Campaign`) and click *OK*.

   In the *Solution Explorer*, an `.xbo` file ( ) appears.
7. Right-click the `.xbo` file and select *Create Script Files*.
   The *Create Script Files* dialog opens.
8. Select `Event: After Modify` and click *OK*.

   An `.absl`file ( ) appears in the *Solution Explorer*.
9. Click *Add New Item* again.
   The *Add New Item* dialog opens.
10. Select *SAP* > *Extension* > *References to Customer-Specific Fields*, enter a name and click *Add*.

    A `.ref` file ( ) appears in the *Solution Explorer*.
11. In the table, select the *Duration* field and click *Save and Activate*.

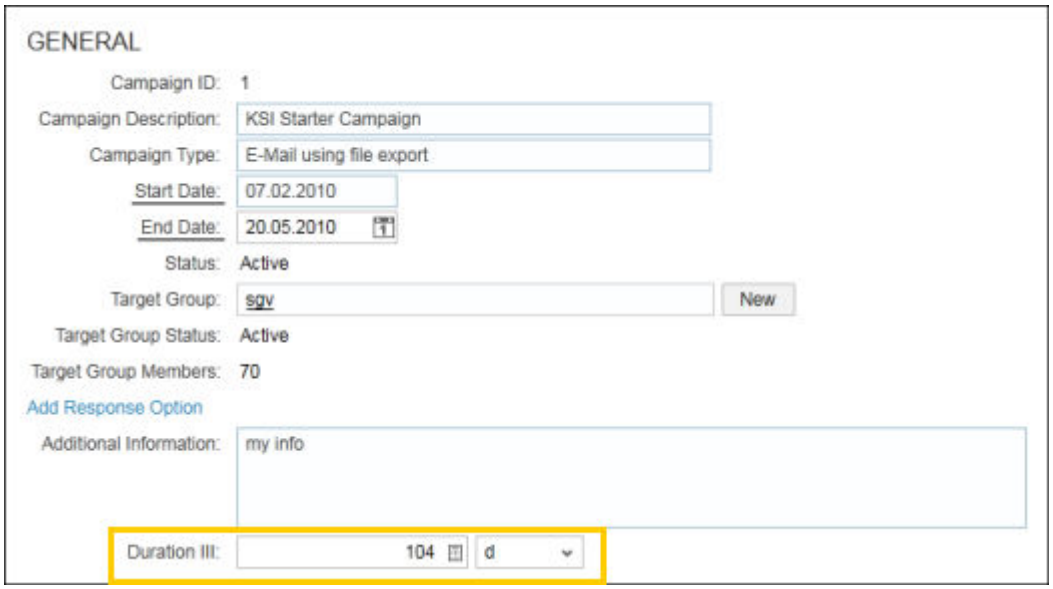
12. Open the `.absl` file and enter the following code:

```
import ABSL;
this.Duration.content =
this.PlannedStartDate.Delta(this.PlannedEndDate).ConvertToDays();
this.Duration.unitCode = "DAY";
```

13. Save and activate the solution.

Test the referenced field in the cloud solution.

1. Log on to the cloud solution and choose *Marketing* > *Campaign Management* > *Campaigns* and click *Edit*.
2. Change the *Planned End Date*.
   The system calculates the duration of the campaign, and the result is displayed in the *Duration* field.



Calculated Duration

Related Information

Create References to Customer-Specific Fields [page 453]

8.4.2 Process Extension Scenarios

8.4.2.1 Process Extension Scenario Quick Guide

A process extension scenario links the data from one business context to other related business contexts.

You create a process extension scenario from a list of predefined extension scenarios. For example, you can create a process extension scenario that links the following business contexts:

- *Account - General Information to Lead - General Information*
- *Lead - General Information to Opportunity - General Information*

Each process extension scenario contains one or more data flows. Each data flow consists of a source and target business context. This reflects how the data is passed from one business context to the next as part of a business process.

For a list of predefined extension scenarios, see Extension Scenarios [page 460].

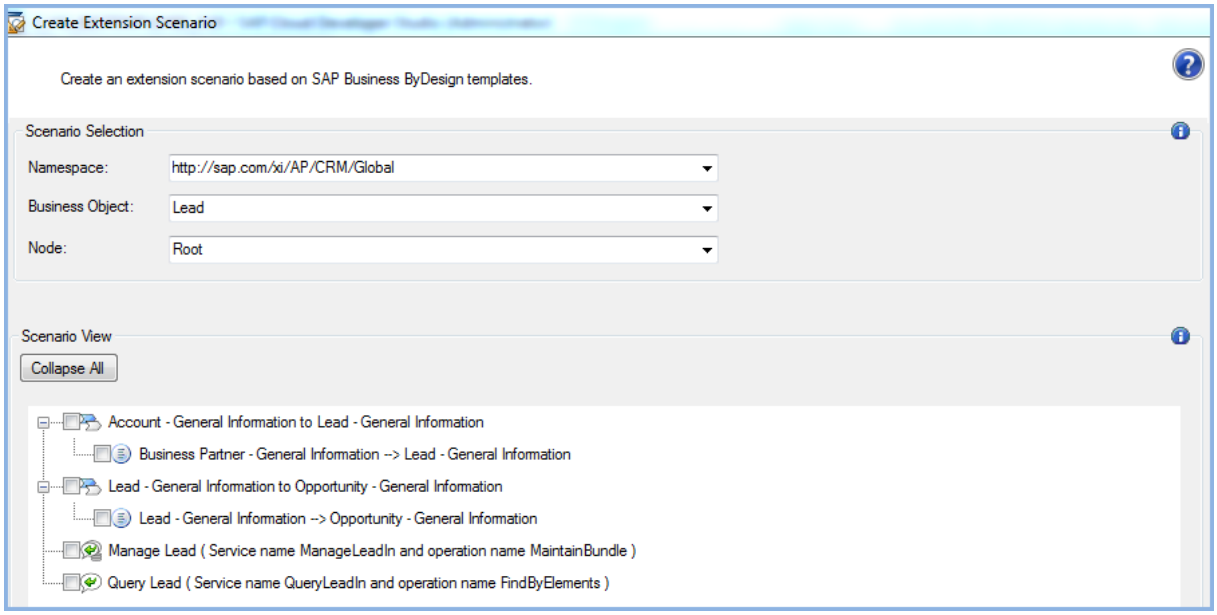
If you now extend any of the business objects involved in the scenario, you can use the annotation `[Scenario]` to declare that the element is available to all the business contexts specified in the scenario.

For example, if you add an element to the Lead business object, you can use an extension scenario to make this element available if you extend the Opportunity. When an Opportunity is created from the Lead, the data for your extension field will be passed to the Opportunity.

B2B Message Extension

You can also extend B2B messages in a process extension scenario. If a predefined extension scenario includes an inbound or outbound B2B data flow, you can create a process extension scenario that includes this data flow. Once an extension field is associated to a scenario that includes such a flow, the extension field value will be propagated to the B2B message during runtime.

When selecting the data flows to include in a process extension scenario, you can identify outbound and inbound B2B messages by their individual icons.



Tasks

Create a Process Extension Scenario

For more information about creating a process extension scenario, see here [page 461].

Add an Extension Field to a Process Extension Scenario

For more information about adding an extension field to a process extension scenario, see here [page 462].

8.4.2.2 Extension Scenarios

Overview

Extension scenarios enable you to create an extension field in one business context and then add that extension field to multiple other business contexts. The data stored in the extension field is then passed from one business context to the next as part of a business process.

The following extension scenarios have been predefined by SAP:

- Account - General Information to Lead - General Information
- Account - General Information to Opportunity - General Information
- Lead - General Information to Opportunity - General Information

8.4.2.3 Create a Process Extension Scenario

Overview

You can create a process extension scenario to link the data from one business context to other related business contexts.

Prerequisites

You have an open solution in the studio.

Procedure

1. In the *Solution Explorer*, right-click your project and select *Add* > *New Item*.
   The *Add New Item* dialog box is displayed.
2. Under *Installed Templates*, click *SAP* > *Extension*.
3. Select the *Process Extension Scenario* template, enter a *Name* for the process extension scenario, then click *Add*.
   The *Extension Scenario Creation* dialog box is displayed.

   > → Tip
   >
   > If the *Create Extension Scenario* window does not open completely, you can choose one of the following two options:
   >
   > - Enlarge the window by clicking and dragging the window frame.
   > - Switch to the default screen resolution. For example, in Microsoft Windows 7, select *Control Panel* > *Display* > *Smaller – 100% (default)*.

4. Select the *Namespace*, *Business Object*, and *Node*.
   Based on your selection, a list of possible extension scenarios is displayed. The individual data flows are displayed under each extension scenario.
5. Select one or more extension scenarios.
   Based on your selection, additional extension scenarios may be displayed.
6. Select additional extension scenarios, as desired.
7. Click *OK*.
   A process extension scenario item with the file extension `.xs` is added to your project.

Related Information

Add an Extension Field to a Process Extension Scenario [page 462]

8.4.2.4 Add an Extension Field to a Process Extension Scenario

Overview

You can use the annotation `[Scenario]` to declare that an element should be available to all the business contexts specified in a process extension scenario.

For example, if you have a process extension scenario that links the Lead, Opportunity and Sales Order business contexts, you can add an element to the Lead business object, and this element will be available if you extended a screen for an Opportunity, a Sales Quote, or a Sales Order.

Prerequisites

- You have an open solution in the studio.
- You have created a process extension scenario as part of your solution.

Procedure

1. In the *Solution Explorer*, right-click your project and select *Add* > *New Item*.
   The *Add New Item* dialog box is displayed.
2. Under *Installed Templates*, click *SAP* > *Extension*.
3. Select the *Business Object Extension* template, enter a *Name* for your business object extension, then click *Add*.
   The *Business Object Selection* dialog box is displayed.
4. Select the *Namespace* of the business object that you want to extend.
5. Select the *Business Object* that you want to extend.
6. Click *OK*.
   A business object extension item with the file extension `.xbo` is added to your project. This file contains a template for the business object extension, including the extendable nodes.
7. In the *Solution Explorer*, double-click the business object extension to open the file in the editing pane.
8. To add extension fields, add elements to the extendable nodes with the scripting language.
9. To include an element in an extension scenario, use the following syntax:

```
[Scenario (Lead_to_Opportunity)] element ext_field_Opportunity_root:Text;
```

> **i Note**
>
> Process extension scenarios are specific to the node for which they were created. If an extension scenario was created for the `root` node, then it can only be called at the `root` node within the `.xbo`. Similarly, a process extension scenario created for the `item` node can only be called from the `item` node within the `.xbo`.

10. Click the *Save* button.
11. Right-click the business object extension and select *Activate*.

Related Information

Scripting Language [page 189]
Business Object Extensions Quick Guide [page 438]
Create a Process Extension Scenario [page 461]

8.4.3 Create an ODATA Service Extension

Overview

You can enhance the existing standard PSM enabled ODATA services with extension fields. You can add the extension fields to ODATA services that are based on the same business object.

Prerequisites

- You have an open In-Development solution in the studio
- The Business Object extension is already active

Process Flow

1. In the Solution Explorer, expand your solution, right-click your project file, and select *Add* > *New Item*. The *Add New Item* dialog box is displayed.
2. Under *Installed Templates* > *SAP*, select *OData Extension*.
3. Enter a *Name* for the OData service extension, and then click *Add*. The *Select Standard OData Service* dialog box is displayed.
4. Enter the *OData Extension Name*.

   > **i Note**
   >
   > You cannot use a name that is already used for a business object or business object extension in your solution.

5. In the *OData Service* dropdown, the system shows the standard OData services delivered by SAP. Select a OData service for which you have enhanced the standard Business Object.
6. Click *OK*. The system opens the *OData Editor* in your default internet browser.

> **Note**
>
> If you are not already logged on to the SAP Cloud Solution, you will be prompted to log on. Use your Cloud Applications Studio user to login to the system.

The OData Editor has two parts. The right pane OData Service is a hierarchical tree that displays all the entity types in the service, and the left pane displays the business object details.

7. Under *OData Service* , select the checkbox for the required entity type. The system displays the Business Object details of the selected entity type on the left pane.
8. Expand the relevant node and select the extension fields that you want to add to the OData Service. The PDI extension fields will be in edit mode and you can select or deselect them. Note that you cannot edit standard SAP fields.
   The selected fields will be added to the entity type.
9. On the right pane, expand the entity type to see the newly added custom field. Select this field to check its properties. You have the option to change few properties, such as, Name and Default value. Note that property names in an entity type should be unique.
10. Once you have added the fields, click *Save* and then click *Activate*.
    You can also activate the OData service from the Cloud Applications Studio. To do this refresh your solution to see a new item with .xodata extension. Right click on the newly added item, and select *Activate*.
    You can also perform the clean, delete, and check operations from the studio.

> **Note**
>
> The same OData service cannot be added more than once in a solution. For example, once you add the Opportunity OData service to a solution, it is not possible to add the same OData service to the solution.

> **Caution**
>
> When you change the name or data type of extension fields in business object extension(.xbo), you must adjust the OData service by removing and adding the property.

Maintenance Mode

You can perform the following actions in a OData Service in maintenance mode:

- **Edit**: Add and remove extension fields from the OData Editor
- **Delete**: Delete OData service extension from the solution

8.4.4 Enhancement Implementations

8.4.4.1 Enhancement Implementations Quick Guide

SAP provides enhancement options based on SAP technology that allow you to enhance the standard behavior of specific applications of the SAP cloud solution without modifying the standard solution. In the studio, you can create enhancement implementations of SAP enhancement options. For example, you can implement a

custom variant of a calculation or override the default filter values to meet country-specific, industry-specific, or customer-specific requirements. You can define your own business logic, for example, for determining, validating or mapping data.

Business and Technical Background

SAP Enhancement Options

An enhancement option is a repository object provided by SAP for a specific application to allow the standard behavior of the application to be enhanced without modifying the standard solution. Enhancement options are defined with metadata and documentation. A distinction is made between single-use enhancement options and multiple-use enhancement options. If an enhancement option is for single use, it can be implemented only once in a solution. If an enhancement option is for multiple use, it can be implemented more than once in a solution.

The documentation of the enhancement options describes how you can use an enhancement option, when the enhancement option is called, and what it does. Furthermore, it gives information about the business configuration content you need to develop for an enhancement implementation, the scoping requirements in the customer system, as well as information about the filters, the input parameters and output parameters, and testing activities.

You can explore SAP enhancement options and view the documentation of the enhancement options using the *Repository Explorer* tool window. For more information, see Repository Explorer [page 49].

Tasks

Create a Business Configuration Set

You can create a business configuration set (BC set) using an SAP business configuration object (BCO) to configure the behavior of standard business processes in the SAP cloud solution. If you create an implementation of an enhancement option which is for single use only, you can also define a BC set using an SAP BCO so that you can enhance the filter values.

For more information, see Create a BC Set Using an SAP BCO [page 383].

Create an Enhancement Implementation

You can create an enhancement implementation of an SAP enhancement option to allow customers to apply business rules that meet specific requirements, for example, country-specific requirements, without modifying the standard functionality of the SAP cloud solution. An enhancement implementation consists of a message definition, a filter, and a script file.

For more information, see Create an Enhancement Implementation [page 466].

Test an Enhancement Implementation

To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the *Business Configuration* work center. For more information, see User Setup Quick Guide [page 33].

8.4.4.2 Create an Enhancement Implementation

Overview

You can create an enhancement implementation of an SAP enhancement option to allow customers to apply business rules that meet specific requirements, for example, country-specific requirements, without modifying the standard functionality of the SAP cloud solution. An enhancement implementation consists of a message definition, a filter, and a script file.

Prerequisites

- You have created and opened a solution in the studio. For more information, see Create a Customer-Specific Solution [page 177].
- If the enhancement option for which you created an implementation is for single use only, you have defined a business configuration set (BC set) using an SAP business configuration object (BCO) so that you can enhance the filter with the value that you want to use in your filter definition. You must also assign the BC set to a business option.
  For information about which SAP BCO is related to which filter, see the public solution model (PSM) documentation for the enhancement option. For information about creating a BC set, see Create a BC Set Using an SAP BCO [page 383].

Procedure

1. In the *Solution Explorer*, click the *Add New Item* button.
   The *Add New Item* dialog box opens.
2. In the *Add New Item* dialog box, select the *Enhancement Implementation* template, enter a name for your enhancement implementation, and then click *Add*.
   The *Create Enhancement Implementation* dialog box opens.

   > **i Note**
   >
   > You must specify a name that begins with an uppercase letter (A-Z) and only contains alphanumeric characters (A-Z, a-z, 0–9) and underscore characters (_). The name must not exceed 120 characters.

3. In the *Create Enhancement Implementation* dialog box, select a namespace and an enhancement option. Optionally, you can select a namespace and business object to filter the list of enhancement options.

A new item with the file extension `.enht` appears in the *Solution Explorer*. By default, the following items are created under the `.enht` file:

- Filter
  An item with the same name as the enhancement implementation and the file extension `.fltr`
- Script file
  An item with the file extension `.absl`, for example, `CONVERT.absl`

The `.enht` file opens in a document window.

4. Under *Message Definition*, define messages for your enhancement implementation by doing the following:
   1. Specify a message ID.
      Each message you define must have a unique ID.
   2. Define the messages parameters one-by-one, from left to right, by doing the following:
      1. In the parameter cell, double-click *...*.
         The *Parameter Selection* dialog box opens. By default, the `http://sap.com/xi/AP/Common/GDT` is displayed and the *Data Type* field is blank.
      2. Select a namespace and a data type.
   3. Specify one or more message texts.
      You use ampersands (&) in the message text to represent placeholders. The number of placeholders in the text must match the number of parameters. For information about defining message texts, see General Standards and Guidelines for Writing System Message Texts [page 792].
   4. Save your changes; then, in the *Solution Explorer*, right-click the `.enht` file and select *Check In Content*.
      The item is locked for editing.
5. In the *Solution Explorer*, double-click the `.fltr` file.
   The filter editor opens in a document window.
6. Under *Filters*, specify a value for each filter.
   - By default, all filters are disabled except for mandatory filters.
   - To enable a filter, you must select the filter checkbox.
   - If you do not specify a value for a mandatory filter, the system reads the value as blank.
   - If a code list is available for a filter, you can select a value from a dropdown list.
   - If you hover over the filter name, the data type of the filter is displayed as a tooltip.
7. Save your changes; then, in the *Solution Explorer*, right-click the `.fltr` file and select *Check In Content*.
   The item is locked for editing.
8. In the *Solution Explorer*, double-click the `.absl` file.
   The script file opens in a document window. By default, the following code is displayed:

```
import ABSL;
```

```
var result : AEString;
```

```
return result;
```

9. In the code editor, define the business logic for your enhancement implementation.
10. Save your changes; then, in the *Solution Explorer*, right-click the `.absl` file and select *Check In Content*.
    The item is locked for editing.
11. In the *Solution Explorer*, right-click the `.enht` file and select *Activate* to activate the messages, the filters, and the business logic you defined.

Result

You have created and defined messages, filters, and the business logic for your enhancement implementation. You can now test your enhancement implementation in the SAP cloud solution.

8.5 Reuse Libraries

8.5.1 Reuse Library Quick Guide

There are two types of reuse libraries: custom reuse libraries that help you to organize the code in your solution or to reuse code in your script files, and SAP reuse libraries, which offer a variety of predefined functions.

Business background

Custom Reuse Libraries and Custom Reuse Functions

A reuse library is the container for reuse functions, whereas the reuse functions provide the code and specify the parameters. You can define reuse functions, for example, to execute mathematical operations or to read or change data of a business object. You can create and define your own reuse libraries and reuse functions to organize complex code in your solution or to reuse your code in script files. Your reuse libraries are called "custom reuse libraries" to differentiate from the SAP reuse libraries.

Basic SAP Reuse Libraries

These reuse libraries provide basic functions that are used often, for example, to retrieve context data, such as the current date or time for the identity of the user currently logged in.

For more information, see Basic SAP Reuse Libraries Reference [page 274].

SAP Reuse Libraries for Business Areas

These reuse libraries provide additional, very specific functions that are relevant for business areas of SAP cloud solutions, for example, in the Financial Management business area. These functions are based on entities provided by SAP.

For more information, see SAP Reuse Libraries for Business Areas Reference [page 308].

SAP Built-In Functions

The SAP built-in functions extend the scope of the syntax for the implementation of your business logic in a similar way as the basic reuse libraries. However, in contrast to the reuse libraries, you do not need to import a namespace to use built-in functions.

For more information, see SAP Built-In Functions Reference [page 320].

Important tasks

Create Your Reuse Library

1. In the *My Solutions* tool window, open your solution.
2. In the *Solution Explorer*, right-click your solution and select *Add* > *New Item*.
3. In the *Add New Item* dialog box, select *Reuse Library* and click *Add*.
   A new item with the extension `.library` appears in the *Solution Explorer*.
   A document window opens, in which you create new reuse functions.

Create Your Reuse Function

In the studio, you can create a reuse function by adding it to an existing or new custom reuse library in your solution. A reuse function is created in two main steps: First, you define the signature of the function by specifying the parameters; then you define the business logic in a separate script file that the system has automatically created for your reuse function.

For more information, see Create Your Reuse Function [page 470].

Add Functions or Parameters to an Existing Reuse Library

1. In the *Solution Explorer*, double-click the reuse library.
   A document window opens that displays the existing reuse functions and their parameters.
2. Select the reuse function that you want to change or create a new reuse function.
   If you have selected a reuse function, you can change its parameters or add new parameters.
   If you create a new reuse function, the *Library Functions* tool window opens and you proceed as described under Create Your Reuse Function [page 470].
3. After you have made your changes, save the reuse library.

Call Your Reuse Function in a Script File

You call your reuse function in an action, an event, or a validation in the same way you call an SAP reuse function.

> **Example**
>
> In this example, the SAP reuse function `Date.ParseFromString` and the custom reuse function `MyLibrary.CalculateAge` are used. To call the SAP reuse function, you must import the `ABSL` namespace.

```
import ABSL;
var birthday = Date.ParseFromString("1975-05-10");
var age = MyLib.CalculateAge( birthday );
```

Use Your Reuse Function in the UI Designer

If you have created a reuse function in the studio and have selected the *Can Be Used in UI Designer* checkbox, you can use this reuse function to calculate the value of a dedicated field in the user interface designer (UI designer) . For more information, see Configure a Dedicated Field [page 750].

> **Note**
>
> Reuse libraries are called transformations in the UI designer.

> **Tip**
>
> To be able to select your reuse function in the UI designer, you must know the namespace of your solution. In the studio, you can find this information if you select your solution in the *Solution Explorer* and then check the *Properties* window to see the namespace.

8.5.2 Create Your Reuse Function

In the studio, you can create a reuse function by adding it to an existing or new custom reuse library in your solution. A reuse function is created in two main steps: First, you define the signature of the function by specifying the parameters; then you define the business logic in a separate script file that the system has automatically created for your reuse function.

Prerequisites

You have created a reuse library in your solution and you have opened this reuse library.

Procedure

1. In the document window of your reuse library, click *Create Function*.
   The *Library Functions* tool window opens.
2. In the *Create a Reuse Function* step, enter a name for the reuse function. You can also select one of the following options:
   - *Write-Enabled*
     The default setting for reuse functions is that they can be used only to read business object data. However, you can change this setting and enable a reuse function to modify data of business objects in a specific deployment unit.

     > **i Note**
     >
     > Please note the following restrictions for the *Write-Enabled* option:
     > - You cannot call a write-enabled reuse function from a script file that is read-only, for example, a validation.
     > - You cannot use a write-enabled reuse function in the user interface designer (UI designer).

   - *Can Be Used in UI Designer*
     If you want to use your reuse function in the UI designer, you must select this option.

     > **i Note**
     >
     > Please note the following restrictions for the *Can Be Used in UI Designer* option:
     > - If you select this option, you cannot use collections as a parameter type.
     > - You cannot select this option if the reuse function is write-enabled.

3. Click *Next*.
4. In the *Define Parameters* step, do the following to define an import parameter:
   1. In the first line of the table, enter a name for the import parameter.
   2. In the same line, under *Type*, select a data type or a business object node.
      If you have selected a business object node, the *ElementsOf* option is selected automatically, which means that the data structure of this node will be passed on by the reuse function.
   3. In addition, you can specify the following:
      - You can make the parameter optional.
      - You can specify that the data type or the business object type is a collection (`collectionof`).
5. Repeat the last step for each import parameter. You can also define one return parameter.
   You can click *Move Up* and *Move Down* to change the order of the parameters.
6. Click *Next*.
   If some of your import parameters are optional, the system changes the order of your parameters so that the mandatory parameters come first. If you have added an output parameter, the system places this parameter at the end of the list.
7. In the *Review* step, check that the information you have entered is correct and then click *Finish*.
8. Save your reuse function.
   A new item with the file extension `.absl` appears in the *Solution Explorer* below the library file.
9. Open the `.absl` file in the code editor to define the business logic of your reuse function.

   > **i Note**
   >
   > You cannot raise messages in a reuse function.

To hide or display the signature in the script file, press **CTRL** + **E** and then **CTRL** + **S**.

Result

You can call this reuse function in a script file. For more information, see Reuse Libraries Quick Guide [page 468].

8.6 Code List Data Type

8.6.1 Create a Code List Data Type

A static code list contains a list of values and descriptions and is used, for example, to provide dropdown lists on UIs. In the studio, you can define a code list data type for your solution and use it when you create your custom business object, a business object extension, or a business configuration object.

If a customer needs to be able to change the code list values, use configurable code list data types instead. For more information, see Create a Business Configuration Object [page 386].

You can delete the code list data type when your solution is in maintenance mode. For more information, see Maintenance Mode [page 185].

Prerequisite

You have created the solution for which you want to create a code list data type. For more information, see Create a Customer-Specific Solution [page 177].

Procedure

1. In the *Solution Explorer*, select your solution and click *Add Item*.
2. In the *Add New Item* dialog, select *Code List Data Type*.
3. Enter a name and click *Add*.
   The *Code List Data Type Wizard* opens.
4. In the *Create Code List Data* step, proceed as follows:
   1. Enter a name and a description.

      > **i Note**
      >
      > The suffix `Code` is added automatically to the name you enter in the `Name` field.

2. Click *Next*.

5. In the *Define the Code List Values* step, proceed as follows:
   1. For all the code list values you want to define, you must enter a value and a description.
      The *Constant* field is filled automatically with a default value (`CO_[YOUR VALUE]`). You can adjust the constant but it has to start with `CO_`. Constants can be used in script files to access code values: `<CodeListDataType>.<Constant>, for example, SeasonsCode.CO_FALL`.

      > → Tip
      >
      > You can use the tab key to navigate between the fields and rows.

   2. Click *Next*.
6. In the *Review* step, review your entries and click *Finish*.
   In the *Solution Explorer*, a file is added to your solution. It is called `[name of your code list].codelist`.
7. Right-click the `.codelist` file and select *Activate*.

Result

A new data type is generated. You can use it when you create business objects, business object extensions, or business configuration objects. The code list values can be exported for translation. For more information, see Translation Quick Guide [page 651].

Related Information

Data Types [page 92]
Example: Create a Code List Data Type (Purchase Order Request) [page 473]

8.6.2 Example: Create a Code List Data Type (Purchase Order Request)

Use Case

In a company, employees create purchase order requests to order goods and services. In the system, they need to choose the purchase order type: standard purchase order or collective purchase order. If they choose a special offer purchase order, they benefit from a price discount. For a special offer purchase order request, an additional field needs to be maintained, which is checked in a script validation.

Prerequisites

- You have a studio developer user and a business user for the SAP cloud solution. For more information, see User Setup Quick Guide [page 33].
- You have created a solution. For more information, see Create a Customer-Specific Solution [page 177].

In the studio, create the code list data type.

1. Log on to the studio and, in the *Solution Explorer*, select your solution and click *Add Item*.
2. In the *Add New Item* dialog, select *Code List Data Type*.
3. Enter a name (**`PurchaseOrderType`**) and click *Add*.
   The *Code List Data Type Wizard* opens.
4. In the *Create Code List Data* step, proceed as follows:
   1. Enter a name (**`PurchaseOrderType`**; the suffix `Code` is added automatically) and a description (**`Purchase Order Type`**).
   2. Click *Next*.
5. In the *Define the Code List Values* step, proceed as follows:
   1. Enter the following values:

Value	Description
STANDARD	One-Time Purchase Order
COLLECTIVE	Collective Purchase Order
SPECIALOFFER	Special Offer Purchase Order

   2. Click *Next*.
6. In the *Review* step, review your entries and click *Finish*.
7. In the *Solution Explorer*, a file is added to your solution. It is called `PurchaseOrderType.codelist`.
8. Right-click the `PurchaseOrderType.codelist` file and select *Activate*.

In the studio, create a business object.

1. In the *Solution Explorer*, right-click the project in your solution and select *Add New Item*.
2. In the *Add New Item* dialog box, select the *Business Object* template, enter a name for your business object (**`PurchaseOrderRequest.bo`**), and click *Add*.
   A new item (`PurchaseOrderRequest.bo`) appears in the *Solution Explorer* and the business object editor opens.
3. In the editor, enter the following coding:

```
import AP.Common.GDT as apCommonGDT;
businessobject PurchaseOrderRequest raises SpecialOfferMessage {
[Label("Purchase Order Request ID")] [AlternativeKey]element
PurchaseOrderRequestID : ID;
[Label("Purchase Order Type")] element PurchaseOrderType :
PurchaseOrderTypeCode;
[Label("Account ID")]element AccountID : BusinessPartnerInternalID;
[Label("Product ID")]element ProductID : ProductID;
[Label("Description")]element Description : MEDIUM_Description;
[Label("Price")]element Price : Amount;
[Label("Discount")]element Discount : Percent;
[Label("Requestor")]element Requestor : UUID;
```

```
[Label("Manager")]element Manager : UUID;
[Label("Account Manager")]element AccountManager : UUID;
[Label("Special Offer")]element SpecialOffer : UUID;
message SpecialOfferMessage text "Please add special offer";
```

4. Save and activate the business object.

In the studio, create a script file.

1. Right-click the business object and select *Create Script Files*.
   The *Create Script Files* editor opens.
2. Select the *Validation: On Save* option and click *OK*.
3. In the editor, enter the following coding:

```
import ABSL;
if (this.PurchaseOrderType == PurchaseOrderTypeCode.CO_SPECIALOFFER &&
this.SpecialOffer.IsInitial())
{
 raise SpecialOfferMessage.Create("E");
return false;
}
return true;
```

4. Save and activate the script file.

In the studio, create the screen.

1. Right-click the business object and select *Create Screens*.
2. In the *Create Screens* dialog, select *Quick Activity Floorplan (QAF)* and click *OK*.
   In the Solution Explorer, a new file appears (`PurchaseOrderRequest_QA.QA.uicomponent`).

In the SAP cloud solution, test the code list data type.

1. Right-click the `.uicomponent` file and select *Preview Screen*.
2. Log on to the SAP cloud solution with the business user.
3. The *PurchaseOrderRequest:<ID and/or Description>* screen opens.
4. Under *Purchase Order Type*, check that a dropdown list box is available with the defined codes and descriptions.
5. Test the `Validation: On Save` check you have defined:
   1. Enter a *Purchase Order Request ID*.
   2. Select the *Purchase Order Type* `SPECIALOFFER`.
   3. Save your entries.
      The error message `Please add special offer` should be displayed.

Related Information

Create a Code List Data Type [page 472]

8.7 Data Protection and Privacy

8.7.1 Enable Data Disclosure

Custom BOs and BO extensions (XBOs) that may have personal data is available in the *Data Protection and Privacy* work center. This section gives you a step-by-step guide as to how you can enable the data disclosure in the user interface.

Prerequisites

You have access to the following work centers:

- *Business Partner Data*
- *Account Management*
- *Data Protection and Privacy*

Configure Business Object and UI

1. Log on to the **SAP Cloud Applications Studio** and choose your system.
2. Go to *My Solutions* and create a new solution.
3. Create a Business Object.
4. Create an association to the business partner object that you want to configure using the following sample code:

> **i** Note
>
> In this sample, we will create an association to Employee.

```
import AP.Common.GDT as apCommonGDT;
import AP.FO.BusinessPartner.Global;
```

```
businessobject EmpAdditionalData {
```

```
[Label("Employee ID")] [AlternativeKey] element  EmployeeID : EmployeeID;
```

```
node SocialData[0,n]{
  [Label("Social Media")] element socialmedia : SocialMediaCode;
  [Label("Handler")] element handler : LANGUAGEINDEPENDENT_SHORT_Name;
}
```

```
node CommunicationData[0,n]{
  [Label("Comm. Media")] element commmedia : CommDataCode;
  [Label("Detail")] element detail : LANGUAGEINDEPENDENT_SHORT_Name;
}
```

```
node AddhocData[0,n]{
  [Label("Name")] element namecode : AddhocCode;
  [Label("Detail")] element detail : LANGUAGEINDEPENDENT_SHORT_Name;
}
```

```
 association ToEmployee to Employee;
```

```
}
```

5. Create a Script file, for example, **AfterModify** to fill the data for this BO.

```
import  ABSL;
```

```
import AP.FO.BusinessPartner.Global;
```

```
var  emproot;
var  empid;
var  empquery;
var  selparam;
var  selresult;
var  empUUID ;
```

```
import AP.FO.BusinessPartner.Global;
if (!this . EmployeeID.IsInitial()){
  empid = this.EmployeeID.content;
  empquery = Employee.QueryByIdentification;
  selparam = empquery.CreateSelectionParams();
  selparam.Add(empquery.InternalID,"I", "EQ",empid);
  selresult = empquery.ExecuteDataOnly(selparam);
  empuuid = selresult.GetFirst().UUID;
  this.ToEmployee = Employee.Retrieve(empuuid);
}
```

Create Screens for the Business Object

1. Under *Project Explorer*, right-click your BO and select *Create Screens*.
2. In the *Create Screens* wizard, enter the *Screen Name* and *Supported Devices*.
3. Under *Screen Creation*, select the *Screen Scenario with Thing-Based Navigation* radio button provide a short ID.
   The Object worklist (OWL) would act as a navigation target for the corresponding disclosure entity of this BO.
4. Click *OK*. The generated screens are displayed in the solution explorer.
5. Open the OWL in the UI Designer.
6. Go to the *DataModel* tab and add data field to the search parameter.
7. Bind the datafield to the SAP_ToEmployee association.
8. Navigate to the *Controller* tab and select the *OBN Inport* checkbox.
9. Under *OBN Configuration* select the *BO Based* radio button.
10. Enter the following details in the available fields on the screen:

Field Name	User Entry
Select Namespace	*http://sap.com/xi/AP/FO/BusinessPartner/Global*
Select BOmodel	*Employee* **i Note** In this example, we have used Employee as the business partner, therefore Employee BO is selected.
Select Operation	*ObligationToDisclose*
Context Attribute	Provide an OBN context that starts with the SolutionPrefix.
PortType Package	*/SAP_BYD_APPLICATION_UI/DataPrivacy/Reuse/EE_OBLIGATION_TO_DISCLOSE_PTP.PTP.uicomponent*
PortType Reference	*Obligation_to_disclose*

11. Map the Employee UUID in the parameters to the data field which is supposed to receive the Employee UUID.
If you are using any other profiles, you need to use the particular combination for BO/PTP and operation.

	Business ByDesign	SAP Hybris Cloud for Customer
Employee	○ BO – Employee ○ PTP – /SAP_BYD_APPLICATION_UI/DataPrivacy/Reuse/EE_Obligation_to_disclose_PTP.PTP.uicomponent ○ Operation – ObligationToDisclose	○ BO – Employee ○ PTP – /SAP_BYD_APPLICATION_UI/DataPrivacy/Reuse/EE_Obligation_to_disclose_PTP.PTP.uicomponent ○ Operation – ObligationToDisclose
Service Agent	○ BO – Employee ○ PTP – /SAP_BYD_APPLICATION_UI/DataPrivacy/Reuse/SA_Obligation_to_disclose_PTP.PTP.uicomponent ○ Operation – SAObligationToDisclose	
Private Account	○ BO – Business Partner ○ PTP – /SAP_BYD_APPLICATION_UI/DataPrivacy/Reuse/ObligationToDisclosureGenericPTP.PTP.uicomponent ○ Operation – ObligationToDisclose	○ BO – Business Partner ○ PTP – /SAP_BYD_APPLICATION_UI/DataPrivacy/Reuse/ObligationToDisclosureGenericPTP.PTP.uicomponent ○ Operation – ObligationToDisclose

Contact Person	○ BO – Business Partner ○ PTP – /SAP_BYD_APPLICATION_UI/DataPrivacy/Reuse/ObligationToDisclosureGenericPTP.PTP.uicomponent ○ Operation – ObligationToDisclose

12. Configure the OWL query by including Employee UUID as a parameter. The query that you use should have Employee UUID as a parameter. The generated BO should by default have query by elements.

Configure OWL in Personal Data Disclosure Work Center View

1. In the *Configuration Explorer* tab, open */SAP_BYD_APPLICATION_UI/DataPrivacy/Reuse/PersonalDataDisclosure_WCVIEW.WCVIEW.uiwocview*
2. In the *Extensibility Explorer* tab, click *Assign Component to View*
3. In the *Add Assigned Object* window, select the type as NavigationTarget , and under *Component*, select the OWL you have created.
4. Save and activate the OWL.

Create New Content Type in the Studio

1. Right click the solution, click *Add* > *New Item* and under *Installed Templates*, select *Data Protection and Privacy* > *Personal Data Disclosure* .
2. Click *Add*. The system opens the personal data disclosure wizard.
3. In the *Data Disclosure Profile* field, select the profile for which you want to create personal data disclosure. In this example, choose Employee and click *Next*.
4. In the wizard that appears, enter the following details:
   - *Disclosure Name*: This is the name that is displayed on the UI.
   - *Disclosure Description*
   - *UI Screen*: Select the OWL that you created in the previous step. This is the navigation target for this disclosure entity.
   - *Inport*: Select the OBN inport that you configured in the previous step.
   - *Determination Functions*: Make sure that the *Count* checkbox is selected. This is used to show the number of relevant records for the entity.
5. Click *Add*. The values are displayed in the table at the bottom of the screen.
6. Click *Next* and check for the information you just filled and click *Finish*.
   The *GET_RECORDS.absl* file is generated.
7. Write the following logic to calculate the number of records:

```
import AP.PC.InformationLifeCycleManagement.Global;
import AP.FO.BusinessPartner.Global;
```

```
var result:DataDiscloseBAdIDestroyCustomDataChangingStructure;
```

```
var  emproot;
var  bpid;
var  empquery;
var  selparam;
var  selresult;
var  empUUID =  InputData.EmployeeUUID;
```

```
emproot = Employee.Retrieve(empUUID);
bpid = emproot.Identification.GetFirst().BusinessPartnerID;
empquery = EmpAdditionalData.QueryByElements;
selparam = empquery.CreateSelectionParams();
selparam.Add(empquery.EmployeeID.content, "I", "EQ",bpid);
selresult = empquery.ExecuteDataOnly(selparam);
```

```
result.RecordsCountValue = selresult.Count();
return  result;
```

8. Under *Project Explorer*, select *GET.RECORDS.absl* > *Activate* .
9. Similarly, select the .pdd file and click *Activate*.
   The custom business object is available in the *Data Protection and Privacy* work center.

Verify Custom Business Objects Record on the UI

1. Log on to the SAP system as an administrator.
2. Navigate to *Data Protection and Privacy* > *Personal Data Disclosure* .
3. Select an employee and click *Disclose Data*. The system opens the *Disclose Employee Data* window.
4. Under *Records*, the system displays the number of records for the custom business object.
5. Click the custom business object to navigate to a new window. The system displays all the relevant data for the employee.

8.7.2 Enable Data Removal

Custom BOs and BO extensions (XBOs) that may have personal data must be removed or anonymized when the person's details are removed using the data privacy tool. This section explains how you can remove or anonymize such data using the SAP Cloud Applications Studio.

You can remove personal data for the following types of business partners:

- Employees
- Service Agents
- Individual Customers
- Contacts

Prerequisites

You have access to the following workcenters:

- *Business Partner Data*
- *Account Management*
- *Data Protection and Privacy*

Configure Business Object and UI

1. Log on to the **SAP Cloud Applications Studio** and choose your system.
2. Go to *My Solutions* and create a new solution.
3. Create a Business Object.
4. Create elements of the Business Object where you plan to store personal data. Here's a sample code:

```
import AP.Common.GDT as apCommonGDT;
```

```
businessobject PrivateAccBank {
```

```
[Label("Account ID")][AlternativeKey] element AccountID :
BusinessPartnerInternalID;
[Label("Description")] element description :
LANGUAGEINDEPENDENT_MEDIUM_Description;
```

```
node BankDetail[0,n]{
  [Label ("Benefits Account No.")] element benefitsacc : IntegerValue;
  [Label ("Bank Account No.")] element bankaccno : BankAccountID;
  [Label ("Bank Name")] element bankname : LANGUAGEINDEPENDENT_Name;
  [Label ("Bank Country")] element bankcountry : CountryCode;
}
```

```
}
```

5. Generate screens.

Create a Personal Data Destruction Enhancement Implementation

1. Right click the solution, click *Add* > *New Item* and select *Extension* > *Enhancement Implementation*.
2. Enter a name and click *Add*. The *Create Enhancement Implementation* wizard opens.
3. In the *Namespace* field, select *http://sap.com/xi/AP/PC/InformationLifecycleManagement/Global*.
4. In the *Enhancement Option* field, select *DataPrivacyDestructionofCustomPersonalData*.
5. Click *OK*. The system creates three files with .enht, .absl. and .fltr extensions.
6. Double-click the .enht file and define the message you want to display on the UI. To do this, enter the *ID* and *Text* under the *Message Definition* header. Do not define any personal data other than the Employee ID in the message parameters.
7. Open the DESTROY.absl file and add a script to remove or anonymize personal data stored in PDI custom BOs and PDI extension fields in XBOs.

```
Disclaimer:
```

```
  The script is write-enabled. Therefore, it should be used for personal data
deletion and anonymization only.
  Creating or modifying standard business object nodes and fields, and
calling actions and web services is strictly prohibited.
```

```
import  AP.PC.InformationLifeCycleManagement.Global;
import  AP.FO.BusinessPartner.Global;
import  AP.PDI.ABSL;
```

```
var  result : DataDestructionBAdIDestroyCustomDataResultDataStructure;
var  custroot;
var  CustomerID;
var  PrivateAccBankbo;
var  currenttime;
var timestr;
```

```
if(InputData.BusinessPartnerRole == "Private Account"){
  custroot = Customer.Retrieve(InputData.BusinessPartnerUUID);
  CustomerID = custroot.InternalID;
  PrivateAccBankbo = PrivateAccBank.Retrieve(CustomerID);
  currenttime = Context.GetCurrentSystemTime();
  timestr = currenttime.ToString();
  PrivateAccBankbo.AccountID = "XXXX".Concatenate(timestr);
  foreach(var  bankdetailnode in  PrivateAccBankbo.BankDetail){
    bankdetailnode.bankaccno = "XXXX".Concatenate(timestr);;
    bankdetailnode.benefitsacc = 9999999;
  }
```

```
  raise DataAnonymized.Create("I");
  result.ErrorOccurredDuringDestroy = false;
}
```

```
return  result;
```

8. Save and activate the enhancement file.

Execute Data Removal on the User Interface

1. Log on to the SAP system as an administrator.
2. Navigate to *Data Protection and Privacy* > *Personal Data Removal*.
3. Select an employee, service agent, individual customer, or private account and click *Remove Data*. The corresponding window opens. For example, if you select an employee, the *Remove Employee Data* window opens.
4. Click *Delete*, and then click *Close*. A background job is created with a corresponding Run ID.

> **i Note**
>
> The system runs a check and gives you an error if the selected data is within the retention period.

5. Navigate to *Data Protection and Privacy* > *Common Tasks*, and click *Administer Data Removal Runs*. The system displays a list of all Data Removal Runs. The generated Run ID will be in *Active* status.
6. Continue refreshing your screen until the *Removal Failed* flag turns to *No*.
   The custom BO instances are anonymized.

8.7.3 Enable Read Access Logging

Overview

You can log and monitor read access to sensitive personal data, such as bank data, using Read Access Logging (RAL). You can define fields that contain sensitive personal data for business partners (such as, Employees, Service Agents, Individual Customers, and Contacts), and custom BO fields.

Whenever these fields are viewed by an end user, a Read Access Log (RAL) entry is created. Fields can be defined as sensitive personal data only if the BO or node is enabled for Change history.

A RAL field group is created that contains all the sensitive personal fields defined in the solution.

Read Access Logging is enabled for the following channel/medium of access:

- Query (Standard and Custom)
- Web service (Read and Query Operations)
- Analytics Datasource and Reports
- ABSL Scripts

> **i Note**
>
> You are not allowed to:
>
> - Debug or trace the PDI solution in the production system, if RAL is scoped and any RAL field group is active.
>   However, if you want to debug the solution, your administrator must assign your user to the *Production Debugging Authorization* work center view. After the debugging is complete, it is recommended that the authorization is removed.
> - Use sensitive personal data fields as placeholders in workflow rules.
> - Use custom queries in enterprise search if they contain sensitive personal data fields.

Prerequisites

- You have created and opened a solution in the studio. For more information, see Create a Solution [page 177].
- You have ensured that the solution contains Custom BO nodes are annotated with ChangeHistory. For more information, see Change History (Business Object) [page 199]
- You have ensured that the solution contains Custom BO nodes that may have sensitive fields have association to BusinessPartnerProjection BO (Customer, Employee or BusinessPartner). For more information, see Association (Business Object) [page 196]

Process Flow

Create New Content Type in the Studio

1. Right click the solution, click *Add New Item* and under *Installed Templates*, select *Data Protection and Privacy* > *Sensitive Personal Data*.
2. Add a name for the file.
3. Click *Add*. The *Sensitive Personal Data Wizard* opens.
4. In the *Select Data Protection and Privacy Level* step, select *Sensitive Personal Data* and click *Next*.
   In the *Sensitive Personal Data Field Selection* step, the *Field Group Name* is automatically filled as the solution name.
5. On this screen, you have an option to add fields from Standard BOs and Custom BOs. Under *Custom BO Fields*, click *Add*.
   The *Select Sensitive Personal Data Field* popup opens.
6. Enter a unique *Item Description*.
7. Select a Business Object from the dropdown. Note that the dropdown shows only custom BOs created in the solution.
8. Select a node. Note that the dropdown shows only nodes that have ChangeHistory annotation.
9. In the *Select Field* tab, choose a field that you want to mark as sensitive personal data.
10. In the *Select ID* tab, choose an association with the Business Partner from the available list.

    > **i** Note
    >
    > - You can only select *InternalID* or *EmployeeID* field. These fields are meant to identify the Business Partner whose data has been read.
    > - For extension BOs, this tab is not available.

11. Click *OK*. A row is added to the *Custom BO Fields* table.
    Follow the steps above to add more fields.
12. Optional: Add a field under *Extension Fields* following the same steps as above.
    The field you add will also be added for other projections of the Business Partner. For example, if you define an extension field in a **Contact** BO, it will be available in **Individual Customer** BO also. The system displays it on screen under the *BO Name* field.
13. Click *Next*.
14. In the *Review* step, check that the information you have entered is correct and then click *Finish*.
    A new item with the file extension .psd appears in the *Solution Explorer*.
15. In the *Solution Explorer*, right-click the file and select *Activate*.
    A Field Group with the same name as the solution is created. The field group is available in the *Data Protection and Privacy* work center.

> **i** Note
>
> - Each solution can have only one Sensitive Personal Data file.
> - You cannot select fields from Dependent Objects.
> - If you have marked an extension field as sensitive personal data, you cannot mark it as personal data.
> - You cannot delete sensitive personal data fields from a solution in maintenance mode.

Verify Field Group on the User Interface

1. Log on to the SAP system as an administrator.
2. Navigate to *Data Protection and Privacy* > *Read Access Logging* > *Field Group Configuration*.
3. Under *Field Group*, select the RAL field group you created. The corresponding sensitive personal data fields will show up in the *Details* section.

8.7.4 Enable Personal Data for Extension Fields

Overview

You can define fields that contain personal data for business partners in extension BOs. The personal data fields content is destroyed when the corresponding business object instance is removed using the data removal process.

Currently, the extension fields of only business partner, customer, employee, and service agent are allowed to be defined as personal data.

Prerequisites

- You have created and opened a solution in the studio. For more information, see Create a Solution [page 177].
- You have ensured that the solution contains a Business Partner BO and its projection (Employee, Customer and BusinessPartner) with extended field which is ChangeHistory relevant. For more information, see Change History (Business Object) [page 199]

Process Flow

Create New Content Type in the Studio

1. Right click the solution, click *Add New Item* and under *Installed Templates*, select *Data Protection and Privacy* > *Sensitive Personal Data*.
2. Add a name for the file.
3. Click *Add*. The *Sensitive Personal Data Wizard* opens.
4. In the *Select Data Protection and Privacy Level* step, select *Personal Data* and click *Next*.
   In the *Personal Data Field Selection* step, you can select extended fields from standard Business Objects.
5. Under *Extension Fields*, click *Add*.
   The *Select Personal Data Field* popup opens.
6. Select a Business Object from the dropdown. Note that the dropdown shows only extension BOs created in the solution.

7. Select a node. Note that the dropdown shows only nodes that have ChangeHistory annotation.
8. In the *Select Field* tab, choose the fields that you want to mark as personal data.
9. Click *OK*. A row is added to the *Extension Fields* table.
   Follow the steps above to add more fields.
10. Click *Next*.
11. In the *Review* step, check that the information you have entered is correct and then click *Finish*.
    A new item with the file extension .psd appears in the *Solution Explorer*.
12. In the *Solution Explorer*, right-click the file and select *Activate*.

> **i** Note
>
> In the maintenance mode, you cannot perform the following tasks:
>
> - Delete a personal data field from a solution.
> - Remove an already existing personal data field.

8.8 Approval Process

8.8.1 Approval Process Quick Guide

You can use the *Approval Wizard* to create an approval process in the root node of your custom business object.

For more information, see Approval Process [page 493].

Availability

> **⚠** Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

The approval process built-in functions are only used in action script files that are created when you define an approval process for a business object. Each action script file initially contains the call of the relevant built-in function. The built-in functions are used to set the status of the business object and to manage related approval tasks.

For more information, see Approval Process Built-In Functions [page 491].

Important tasks

Create an Approval Process

In the studio, you define one approval process per business object in the root node. An approval process contains at least one approval task and can have notifications and a revision task. For each task type, you define the task recipients by selecting elements from the custom business object. Further, you can select business object elements that an administrator can use to define approval step conditions in the SAP cloud solution.

For more information, see Create an Approval Process [page 495].

Assign Approval Process Tasks to the UI

You assign the approval process tasks to the UI to make the tasks and notifications visible in a work center of the SAP cloud solution.

For more information, see Assign Approval Process Tasks to the UI [page 498].

Test the Approval Process

After assigning the approval process tasks to the UI, you can test the approval process you have created in the SAP cloud solution.

For more information, see Test the Approval Process [page 499].

Example: Create an Approval Process (Price Request)

This example shows how to define a multistep approval process for a custom business object created in the studio.

For more information, see Example: Create an Approval Process (Price Request) [page 487]

8.8.2 Example: Create an Approval Process (Price Request)

This example shows how to define a multistep approval process for a custom business object created in the studio.

> Example
>
> **Example**

A sales representative creates a price request to negotiate the contract pricing for an account. On save, the price request is automatically submitted for approval to the manager of the sales representative. If the value of the price request is below 10,000 US$ or the discount is below 15%, the price request is approved automatically. If the value of the price request is greater than 200,000 US$, there is an additional approval step: The responsible account manager needs to approve the price request as well. After the price request is approved, the sales representative receives a notification.

Prerequisites

- You have a developer user for the studio and a business user in the SAP cloud solution. The *Approval Processes* work center view is assigned to your business user.
  For more information, see the User Setup Quick Guide [page 33].
- You have created a solution.
  For more information, see Create a Solution [page 177].

Studio: Create a solution with a business object

1. In the *Solution Explorer*, click *Add New Item* ( ) and select the `Business Object` template.
2. In the *Name* field, enter **`PriceRequest.bo`** and click *Add*.
   In the *Solution Explorer*, the `PriceRequest.bo` file is displayed.
3. In the editor, enter the following coding:

```
import AP.Common.GDT;
businessobject PriceRequest {
[Label("Price Request ID")] [AlternativeKey] element PriceRequestID : ID;
  [Label("Account ID")]      element AccountID : BusinessPartnerInternalID;
[Label("Product ID")]      element ProductID : ProductID;
[Label("Description")]     element Description : MEDIUM_Description;
[Label("Price")]            element Price : Amount;
[Label("Discount")]        element Discount : Percent;
[Label("Requestor")]       element Requestor : EmployeeID;
  [Label("Manager")]         element Manager : EmployeeID;
[Label("Account Manager")] element AccountManager : EmployeeID;
}
```

4. Right-click the file and click *Save and Activate* ( ).

Studio: Create the approval process

1. Right-click the `PriceRequest.bo` file and click *Create Approval Process*.
   The *Approval Wizard* opens.
2. In the *Define Task Properties* step, the name of the *Approval Status Field* is already entered and the *Approval* checkbox is already selected. Select the *Notification* checkbox and click *Next*.

3. In the *Define Task and Notification Texts* step, enter the following texts and parameter and click *Next*.

Tab Page	Approval Task	Approval Notification	Rejection Notification
Subject Field	Please approve price request `&PRICEREQUEST&`	Your `&PRICEREQUEST&` has been approved	Your `&PRICEREQUEST&` has been rejected
Description Field	Dear colleague, Please approve `&PRICEREQUEST&`. Kind regards,	Dear colleague, Your `&PRICEREQUEST&` has been approved. Kind regards,	Dear colleague, Your `&PRICEREQUEST&` has been rejected. Kind regards,

> → Tip
>
> To enter the parameter, proceed as follows:
>
> 1. Place the cursor at the position you want the parameter to appear and click *Add Parameter*. The *Parameters* dialog opens.
> 2. In the `Parameter` field, enter the name **`PRICEREQUEST`** and click the ... button in the same row.
> 3. Select the business object element `PriceRequestID` and click *OK* twice.

4. In the *Select Task and Notification Recipients* step, enter the following values and click *Next*:

Task Type	Description	Business Object Element	Default
Approval	Manager Approval	Root.Manager	☑
Notification	Requestor Notification	Root.Requestor	
Approval	Account Manager Approval	Root.AccountManager	

> → Tip
>
> To select the business object element, click the ... button, select the business object element and click *OK*.

5. In the *Select Fields for Conditions* step, select the business object elements `Price:Amount` and `Discount:Percent`, then click *Next*.
6. In the *Review* step, click *Finish*.
7. In the *Solution Explorer*, right-click the `ApprovalStatus.approvalprocess` file and click `Activate`.
8. Right-click your solution and select *Deploy Business Configuration*.
   Due to technical background processes, it takes some time until the default approval process appears in the SAP cloud solution.

Studio and UI designer: Create the screens

1. In the *Solution Explorer*, right-click the business object `PriceRequest.bo` and select *Create Screens* and then a screen scenario.
   In the *Short ID* field, enter an ID, for example **`Approval`**, and click *OK*.
   The `.uicomponent` files are generated and displayed in the *Solution Explorer*.
2. Right-click the `PriceRequest_WCView.WCVIEW.uiwocview` file and select *Open in UI Designer*.
   The UI designer opens in edit mode.
3. In the *Property Explorer*, under *RBAMData*, select *AssignedObjects* and click the ... button.
   In the *Add AssignedObjects* dialog, proceed as follows:
   1. Click the *Add Row* button.
   2. In the new row, under *AssignedObject Type*, click the down arrow button and select *TaskListView*.
   3. In the same row, under *ComponentID*, click the ... button.
      The *Select Component* dialog opens.
   4. In the folder for your solution (*[solution name]_MAIN* > *SRC*), select the `PriceRequest–Root–APP1–approval` file and click *OK*.
      Repeat steps a. to d. for the `PriceRequest–Root–APP1–notification` file.
   5. Click the *Add Row* button again.
   6. In the new row, under *AssignedObject Type*, do not change the *Navigation Target* option.
   7. In the same row, under *ComponentID*, click the ... button.
      The *Select Component* dialog opens.
   8. In the folder for your solution (*[solution name]_MAIN* > *SRC*), select the `PriceRequest–Root–APP1.QA` file and click *OK*.
      Repeat steps e) to h) for *NotificationUI* and *RevisionUI* files, if any.
4. Save and activate your changes and close the UI designer.

SAP cloud solution: Test the approval process

1. Log on to the SAP cloud solution using your business user and choose *Application and User Management* > *Approval Processes*.
2. In the *Find* field, enter **`PriceRequest`**, and select your `Default approval process for PriceRequest`, and click *Copy*.
3. In the copied approval process, in the *Process Name* field, enter **`Approval Process for PriceRequest`**.
4. To create the manager approval step, do the following:
   1. In the *Process Step Name*, enter **`Manager Approval`**.
   2. In the *Work Distribution* field, select *Manager Approval*.
   3. Under *Conditions*, click *Add Group*.
      The *Edit Conditions dialog* opens.

4. Enter the following values:

Field	Compare Operator	With	
Price	Greater than or equal to	10,000	USD - US Dollar
Discount	Greater than or equal to	15%	

> → Tip
>
> Click *Add Condition* ( ) to add a new condition.

5. To create the account manager approval step, do the following:
   1. Under *Process Steps*, click *Add Row*
   2. In the *Process Step Name*, enter `Account Manager Approval`.
   3. In the *Work Distribution* field, select *Account Manager Approval*.
   4. Under *Conditions*, click *Add Group*.
      The *Edit Conditions dialog* opens.
   5. Enter the following values:

Field	Compare Operator	With	
Price	Greater than or equal to	200,000	USD - US Dollar

   6. Click *Save and Close*.
6. Click *Edit Validity* to adjust the validity of the approval process and click *Save and Close*.

> i Note
>
> - The *Valid To Date* of the default approval process must be the same as the *Valid From Date* of the copied approval process.
> - The copied approval process must have an **unlimited validity**.

7. Refresh the *Approval Processes* view.
   Once the validity of the copied approval process has started, its status should be *Active* while the status of the default process should be *Ended*.

8.8.3 Business and Technical Background

8.8.3.1 Approval Process Built-In Functions

The approval process built-in functions are only used in action script files that are created when you define an approval process for a business object. Each action script file initially contains the call of the relevant built-in function. The built-in functions are used to set the status of the business object and to manage related approval tasks.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

SendForApproval

Description	Requests an approval of a business object instance. If an approval is required, the system creates an approval task and sends it to the approver.
Example	`var result = this.Function::SendForApproval();` **Result**: Approval status “2” (Not Necessary) or approval status “3” (In Approval)

Approve

Description	Approves a business object instance and its related tasks.
Example	`var result = this.Function::Approve();` **Result**: If this approval task is the last step of an approval process, the result is approval status “4” (Approved), otherwise the approval status is “3” (In Approval).

Reject

Description	Rejects a business object instance and its related tasks.
Example	`this.Function::Reject();` **Result**: Approval status “5” (Rejected).

SendBackForRevision

Description	Sends an approval task back to the requester to revise it.
Example	`this.Function::SendBackForRevision();` **Result**: Approval status “6” (In Revision).

Withdraw

Description	Withdraws an approval request for a business object instance.

Example

```
this.Function::Withdraw();
```

Result: Approval status “7” (Withdrawn).

8.8.3.2 Approval Process

Overview

The studio allows you to create a default approval process for a custom business object. You define at least one approval step and, optionally, approval and rejection notifications as well as a revision task. Once the default approval process has been activated, you assign the tasks to the UI. In the SAP cloud solution, an administrator can then add multiple approval steps, define conditions, and change the validity of the approval process. The participants of the process receive tasks and notifications informing them of required actions or the process status.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Features

You create the approval process in the studio. For more information, see Create an Approval Process [page 495]. When you activate the approval process, the following objects are created and added to the custom business object in the *Solution Explorer*:

- Approval process definition including tasks
- Action script files
- Task execution UIs

Approval Task Types

When you define the approval process, you can create the following tasks:

- **Approval (default — mandatory)**
  Informs the person responsible about a business document requiring approval.

- **Notification (approval/rejection — optional)**
  Informs the employee who started the approval process that the business document has been approved or rejected.
- **Revision (optional)**
  Informs the employee who started the approval process that the person responsible for approval asks for changes to the business document.

Action Script Files

When you activate the approval process, the following action script files are generated:

- `SubmitForApproval`
- `Approve`
- `Reject`
- `SendBackForRevision`
- `WithdrawFromApproval`

The following table shows the different status changes caused by the action script files listed above. The default implementation of these actions uses built-in functions for changing the approval status and managing related approval tasks. You can change the default implementation in the relevant script files, for example, by adding messages or logic.

Approval Process Status	Action	New Approval Process Status
Not Started	`SubmitForApproval`	• In Approval • Approval not Necessary
In Approval	`WithdrawFromApproval`	Withdrawn
	`SendBackForRevision`	In Revision
	`Reject`	Rejected
	`Approve`	• In Approval • In Approval
Withdrawn	`SubmitForApproval`	• In Approval • Approval not Necessary
In Revision	`SubmitForApproval`	• In Approval • Approval not Necessary
Rejected	—	—
Approved	—	—

Built-In Functions

The approval process built-in functions are only used in action script files that are created when you define an approval process for a business object.

For more information, see Approval Process Built-In Functions [page 491].

Task Execution UIs

These screens are generated automatically when you activate the approval process. They are used to display tasks and allow the execution of task-related actions. In a manual step, you assign the approval tasks to the UI to see the tasks and notifications in a work center of the cloud solution. For more information, see Assign Approval Process Tasks to UI [page 498].

Approval Process in the Cloud Solution

The default approval processes you create in the studio always have one approval step, no conditions, and an unlimited validity. In the SAP cloud solution, administrators can copy and adjust the default approval process to meet their specific process requirements, for example:

- Create addition sequential or parallel approval steps
- Control the relevance of approval steps by adding conditions
  A typical example is that an approval step is only required, if an amount defined in the custom business object exceeds a certain value.
- Adjust the validity of an approval process

8.8.4 Tasks

8.8.4.1 Create an Approval Process

In the studio, you define one approval process per business object in the root node. An approval process contains at least one approval task and can have notifications and a revision task. For each task type, you define the task recipients by selecting elements from the custom business object. Further, you can select business object elements that an administrator can use to define approval step conditions in the SAP cloud solution.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Prerequisites

- You have activated the business object for which you want to create the approval process. For more information, see Business Objects Quick Guide [page 398].

- For SAP Cloud for Customer and SAP Cloud for Travel and Expense users, you have created an enterprise query. For more information, see **Step 3** under **Procedure** in Create a Query [page 421].

Procedure

1. In the *Solution Explorer*, right-click the business object and select *Create Approval Process*.

   > **i Note**
   >
   > You can only create one approval process for a custom business object.

   The *Approval Wizard* opens.
2. In the *Define Task Properties* step, do the following:
   1. Enter a name in the *Name of Approved Status Field* field.
      The field is added to the business object during activation and is used to store and display the approval status.

      > **i Note**
      >
      > Do not change the approval status field.

   2. Under *Task Types*, the *Approval* checkbox is preselected.
      If you select *Notification*, a notification is generated to inform the employee who started the approval process (requestor) that the request has been approved or rejected.
      If you select *Revision*, a revision task is generated to inform the requestor that the approver has suggested changes to the request.
   3. Click *Next*.
3. In the *Define Task and Notification Texts* step, proceed as follows:
   Enter the *Subject* and *Description* texts for each task and notification to display to the recipients.
   The description should contain instructions describing how to complete the task. It can contain parameters that will be replaced by values from the business object at run time. Parameters are denoted by an ampersand (&) before and after the parameter name. To include parameters in the text, do the following:
   1. Enter a free text in the *Subject* and *Description* fields and then place the cursor at the position you want the parameter to appear.
   2. Click *Add Parameter*.
      The *Parameters* dialog opens.
   3. In the *Parameter* field, enter a name for the parameter, for example `Approver`.
   4. In the same row, click the ... button.
      The *Select Business Object Element* dialog opens.
   5. Select a business object element that you want to include in the text and click *OK*.
   6. Click *OK* to close the *Parameters* dialog.
      If you have defined a parameter for one of the tabs, you can select it again for the other tabs in the *Parameters* dialog and do not have to define it again.
   7. Repeat the steps above on the other tabs and click *Next*.

> **i Note**
>
> The *Approval Notifications*, *Rejection Notifications*, and *Revision* tabs are only displayed, if you have selected the *Notification* and *Revision* options in the previous step.

4. In the *Select Task and Notification Recipients* step, proceed as follows:
   1. Enter a description of the person responsible and select a business object (BO) element by clicking the ... button. Select one of the BO elements and click *OK*.
      You can select the BO elements from the root node as well as from another node of the business object. In the *BO Element* field, the complete path to the element is displayed.

      > **i Note**
      >
      > - You can select BO elements of type UUID or EmployeeID, or fields generated by associations.
      > - The elements must be taken from the business object for which the approval process is defined. You cannot refer to elements in another business object using an association.

   2. Repeat the steps for the other tasks you created.
      You can define various approval tasks for a multistep approval process but only one approval notification, one rejection notification, and one revision task. If you define several approval steps, one of the approval tasks has to be defined as default.
   3. Click *Next*.
5. In the *Select Fields for Conditions* step, proceed as follows:
   1. Select the business object elements that can later be used by the customer's administrator to define conditions.
   2. Click *Next*.
6. Check the *Review* screen and click *Finish*.
   In the *Solution Explorer*, a process definition with the file extension `.approvalprocess` appears. The system also creates action script files (`.absl`) and `.uicomponent` files.

   > **→ Tip**
   >
   > The system creates one `.uicomponent` file per task type. If, by accident, one or various `.uicomponent` files are deleted, you can recreate them. To do this, right-click the `.approvalprocess` file and select *Create Task Execution UIs*.

7. To activate the approval process, right-click the `.approvalprocess` file and select *Activate*.
8. Right-click your solution and select *Deploy Business Configuration*.

Result

The default approval process is created. Due to technical background processes, it takes some time until the default approval process appears in the SAP cloud solution. In the UI designer, you assign the approval process tasks to the UI. This ensures that the tasks are displayed in the cloud solution and that an administrator can use them to define an approval process. For more information, see Assign Approval Process Tasks to UI [page 498].

8.8.4.2 Assign Approval Process Tasks to UI

You assign the approval process tasks to the UI to make the tasks and notifications visible in a work center of the SAP cloud solution.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Prerequisites

You have created an approval process in the studio for the SAP cloud solutions. For more information, see Create an Approval Process [page 495].

Procedure

1. In the *Solution Explorer*, do the following:
   The `.uicomponent` files are generated and displayed in the *Solution Explorer*.
   1. Right-click the business object, select *Create Screens* and choose to create a screen scenario.
   2. Enter a Short ID to identify your screen in the UI designer and in the SAP cloud solution.
      The Short ID is generated as follows: [Technical name of solution]_[Short ID]_WCVIEW. If you have also created a thing type floorplan, an additional Short ID is generated: [Technical name of solution]_[Short ID]_TT.

   > → Tip
   >
   > Right-click the solution and, in the *Property Explorer* in the *Project File* field, look up the name of the solution and note it down.
2. Right-click the work center view file (`.uiwocview`) and select *Open in UI Designer*.
3. In the *Property Explorer*, under *RBAMData*, select *AssignedObjects*.
4. Click the ... button.
   The *Add AssignedObjects* dialog opens, where you proceed as follows:
   1. Click the *Add Row* button.
   2. In the new row, under *AssignedObject Type*, click the down arrow button and select *TaskListView*.

3. In the same row, under *ComponentID*, click the *...* button.
   The *Select Component* dialog opens.
4. In the folder for your solution (*[solution name]_MAIN* > *SRC*), select the `.approval` file and click *OK*.
   Repeat steps a) to d) for the `.notification` and `.revision` files, if there are any.
5. If you have created a notification and/or revision task, click the *Add Row* button again.
6. In the new row, under *AssignedObject Type*, do not change the *Navigation Target* option.
7. In the same row, under *ComponentID*, click the *...* button.
   The *Select Component* dialog opens.
8. In the folder for your solution (*[solution name]_MAIN* > *SRC*), select the *ApprovalUI (APP1.QA.uicomponent)* file and click *OK*.
   Repeat steps e) to h) for the *NotificationUI* and *RevisionUI* file and close the UI designer.

Result

You can now log on to the SAP cloud solution and see the default approval process you created in the *Approval Processes* view.

> **i** Note
>
> For any user to be able to view the approval process tasks in the UI, you must give the user access rights to the relevant work center to which the tasks are assigned. Once this is done, select *Administration* in the *Solution Explorer*, and click on *Update Authorizations and Access Rights* to refresh the access right permissions.

Related Information

Test the Approval Process [page 499]

8.8.4.3 Test the Approval Process

After assigning the approval process tasks to the UI, you can test the approval process you have created in the SAP cloud solution.

Availability

> **⚠** Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud

solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.

For details, see Feature Matrix [page 27].

Prerequisites

- You have added the approval tasks to the UI. For more information, see Assign Approval Process Tasks to UI [page 498].
- You have a business user in the SAP cloud solution with access rights for the *Application and User Management* work center.

Procedure

1. Log on to the SAP cloud solution using your business user and choose *Application and User Management* > *Approval Processes*.
2. Select your default approval process, which has the same name as your custom business object, and click *Copy*.
3. In the copied approval process, adjust the name in the *Process Name* field.
4. Enter a number in the *Step* field and a name in the *Process Step Name* field.
5. In the *Work Distribution* field, select the person responsible for this task.
6. Repeat the steps to create further approval steps. Click *New* and select *Task*, *Notification*, *Alert*, or *Clarification Request,* as required.

   → Tip

   The clarification request corresponds to the revision task selected in the studio.
7. Under *Conditions*, click *Add Group*, and in the *Edit Conditions* dialog, add one or more conditions.
8. Click *Edit Validity* to adjust the validity of the approval process and click *Save and Close*.

   i Note

   - The *Valid To Date* of the default approval process must be the same as the *Valid From Date* of the copied approval process.
   - The copied approval process must have an **unlimited validity**.

Example

Field	Entry
Process Name	Approval Process
Step	1
Process Step Name	Manager Approval (Approval Step 1)
Work Distribution	Manager (Approver 1)
Step	2
Process Step Name	Account Manager Approval (Approval Step 2)
Work Distribution	Account Manager (Approver 2)

For more information, see the approval process documentation in the SAP cloud solution.

8.9 Notification Rules

8.9.1 Enable Notification Rules

In SAP's cloud solutions, notification rules can be defined by customers to receive a notification in case a certain situation occurs for a business object. Typical examples are status changes of business objects or the exceeding of threshold values. The notifications only serve as information and do not trigger automatic actions or tasks.

If a customer wants to use notification rules with a custom business object, you as a partner need to enable the business object for notification rules in the studio: In the *Notification Rules Wizard*, you select business object fields that can be used as placeholders, conditions, and recipients.

After you have enabled the custom business object for notification rules, your customer can create notification rules for the business object in the cloud solution.

For more information, see the "Notification Rules Quick Guide" in the documentation of the SAP cloud solution.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud

solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.

For details, see Feature Matrix [page 27].

Prerequisites

- You have activated the custom business object for which you want to enable notification rules. For more information, see Business Objects Quick Guide [page 398].

If you want to use thing-based navigation with your solution, the following prerequisites apply:

- You have created a thing-based screen scenario. For more information, see Generate Screens for a Business Object [page 429].
- You have created a query. In the *Query Wizard*, in the *Create a Query* step, you have selected the *Include in Enterprise Search* option. For more information, see Create a Query [page 421].

> **i Note**
>
> To be able to select the *Include in Enterprise Search* option, you need to define a `Date/Date Time` field in your custom business object.

Procedure

1. In the studio, in the *Solution Explorer*, right-click the custom business object and select *Enable Notification Rules*.
   The *Notification Rules Wizard* opens.

   > **i Note**
   >
   > You can use this process only once per custom business object.

2. In the wizard, in the *Define Fields for Placeholders and Conditions* step, do the following:
   1. Select the business object elements to be used as fields for placeholders and conditions in notification rules.

      > **i Note**
      >
      > - For the placeholders, you can use business object elements defined in the root node or in [1,1] associated nodes.
      > - For the conditions, you can use business object elements from the root node and from [1,n] associated nodes.
      > - You can only use data types with the following representation terms:
      >
      > | amount | description | measure | quantity |
      > |---|---|---|---|

code	duration	name	text
date	identifier	numeric	time
date/time	indicator	percent	value

To find out the representation term of a data type, call up the relevant data type in the *Repository Explorer*. For more information, see Look Up the Representation Term of a Data Type [page 96].

2. Click *Next*.

3. In the *Define Labels for Placeholders and Condition Fields* step, you can adapt the field labels. Click *OK*.

> **i Note**
>
> Do not leave the field label blank or use identical labels for different fields; field labels must be unique.

4. In the *Define Recipients* step, select the recipients to be used in notifications:
   1. In the *Description* field, enter a description for the recipient, for example, **`Manager`** or **`Requestor`**.
   2. In the *Business Object Element* field, click the ... button.
   3. Select a business object element that you want to use as recipient and click *OK*.

> **i Note**
>
> - You can select business object elements of type UUID or EmployeeID, or fields generated by associations.
> - The elements must be taken from the business object for which the approval process is defined. You cannot refer to elements in another business object using an association.

   4. Click *Next*.
5. In the *Review* step, review your entries and click *Finish*.
6. In the *Solution Explorer*, a `[business object name].notification` file ( ) is created. Right-click the file and select *Activate*.
7. Right-click your solution and click *Deploy Business Configuration*.

Result

You have enabled the custom business object for notification rules. Your customer can now define a notification rule in the cloud solution. For more information, see the “Notification Rules Quick Guide” in the documentation of the SAP cloud solution.

In maintenance mode, the following restrictions apply:

- Delivered fields cannot be deselected as they may have been used in notification rules by customers.
- Delivered fields cannot be renamed.

Related Information

Example: Enable Notification Rules (Price Request) [page 504]

8.9.2 Example: Enable Notification Rules (Price Request)

Use Case

A sales representative creates a price request to negotiate the contract pricing for an account. If the price request amount exceeds USD 50,000, the responsible manager and account manager want to receive a notification.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Prerequisites

- You have a studio developer user and a business user for the SAP cloud solution. For more information, see User Setup Quick Guide [page 33].
- In the SAP cloud solution, the *Notification Rules* work center view is assigned to your business user.
- You have created a solution. For more information, see Create a Customer-Specific Solution [page 177].

In the studio, create a business object and enable it for notification rules.

1. To create a business object in the studio, in the *My Solutions* window, open your solution.
2. In the *Solution Explorer*, right-click your solution and select *Add New Item*.
3. In the *Add New Item* dialog box, select the *Business Object* template, enter a name for your business object (for example, **`PriceRequest.bo`**), and click *Add*.
   A new item with the file extension `.bo` appears in the *Solution Explorer* and the business object editor opens.
4. In the editor, enter the following coding:

```
import AP.Common.GDT as apCommonGDT;
businessobject PriceRequest {
```

```
[Label("Price Request ID")] [AlternativeKey]element PriceRequestID : ID;
[Label("Account ID")]element AccountID : BusinessPartnerInternalID;
[Label("Product ID")]element ProductID : ProductID;
[Label("Description")]element Description : MEDIUM_Description;
[Label("Price")]element Price : Amount;
[Label("Discount")]element Discount : Percent;
[Label("Requestor")]element Requestor : ID;
[Label("Manager")]element Manager : EmployeeID;
[Label("Account Manager")]element AccountManager : ID;
node Documentation [0,n] {
[Label("Price Type")]element PriceType:PriceDetailLevelCode;
[Label("Price Description")]element Description :MEDIUM_Description;
}
}
```

5. Save and activate the business object.
6. In the *Solution Explorer*, right-click your business object and select *Enable Notification Rules*.
   The *Notification Rules Wizard* opens.
7. In the wizard, in the *Define Fields for Placeholders and Conditions* step, select the following business object elements: `Price Request`, `Product ID`, `Price Amount`, and the `Documentation` node and click *Next*.
8. In the *Define Labels for Placeholders and Condition Fields* step, you can adapt the field labels. Click *OK*.
9. In the *Define Recipients* step, select the recipients to be used in the notification (`Account Manager` and `Manager`):
   1. In the *Description* field, enter a description for the recipients.
   2. In the *Business Object Element* field, click the *...* button.
   3. Select the business object elements and click *OK*, then click *Next*.
10. In the *Review* step, review your entries and click *Finish*.
11. In the *Solution Explorer*, a `[business object name].notification` file ( ) is created. Right-click the file and select *Activate*.
12. Right-click your solution and select *Deploy Business Configuration*.
    You have enabled the custom business object for notification rules. Now, you can define a notification rule in the SAP cloud solution.

In the SAP cloud solution, define a notification rule for the business object.

1. Log on to the SAP cloud solution with a business user. In the *Application and User Management* work center, choose *Notification Rules* and then *New*
   The *Notification Rules* guided activity opens.
2. In the *Enter Basic Data* step, proceed as follows:
   1. Enter a description (for example, **`Price Request`**) and select the business object you created.
   2. Enter a subject for the notification (for example, **`Price Request &1 for Product &2 with Price &3`**).
   3. Define your business object elements as placeholders (**`&1 = Price Request ID, &2 = Product ID, &3 = Price`**) and click *Next*.
3. In the *Define Conditions* step, proceed as follows:
   1. Click *Add Row*.
   2. In the *Edit Condition* dialog, make the following entries:

Field	**`Price`**

Compare Operator	`Greater than`
With	`Value 50,000 USD — US Dollar`

3. Click *OK* and *Next*.

4. In the *Define Recipients* step, proceed as follows:
   1. You can define an employee as recipient of the notification by clicking *Add Employee*.
   2. To select the recipients you defined in the studio (`Manager, Account Manager`), click *Add Determination* and *Next*.
5. In the *Review* step, review your entries and click *Activate*, and then *Finish*.
6. In the *Confirmation step*, you can create a new notification rule or click *Close*.

Result

You have now created a notification rule including conditions and recipients. When the conditions are met, a notification is sent to the recipients.

Related Information

Enable Notification Rules [page 501]

8.10 Print Forms

8.10.1 Print Forms Quick Guide

You can create a print form to output the data from a business object that you created in your solution. You can define the data to include on the form, design the layout of the form, and select a floorplan to link to the form. Print forms are read-only portable document format (PDF) documents that are generated from data stored in the system. In the SAP cloud solution, print forms can be printed, sent as an e-mail attachment, or faxed to business partners.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.

For details, see Feature Matrix [page 27].

Business and Technical Background

Business Objects

As part of your solution, you can create new business objects. For more information, see Business Objects Quick Guide [page 398].

Tools for Form Templates

In the SAP cloud solution, customers use form templates to define the content and layout of documents that can be output from the system. For information about how customers work with the tools for form templates, see the following quick guides in the relevant SAP cloud solution:

- Form Template Maintenance Quick Guide
- Form Template Selection Quick Guide

Tasks

Create a Print Form

You can create a print form to output the data from a business object. You use Adobe LiveCycle Designer to define the data you want to include on the form and design the layout of the form.

For more information about creating a print form, see Create a Print Form [page 508].

Create a Preview Button for a Print Form

You can create a *Preview* button and link it to a print form. Clicking the *Preview* button allows you to preview the form as a PDF file and provides options to either save or print the form.

For more information about creating a preview button, see Create a Preview Button for a Print Form [page 734].

Test a Print Form

You can test the print form in the preview after deploying the business configuration content of your solution.

For more information about testing a form, see Test a Print Form [page 510].

8.10.2 Associating an .xdp File with Adobe LiveCycle Designer

Procedure

1. In Windows, select *Control Panel* > *All Control Panel Items* > *Default Programs*.
2. Select *Associate a file type or protocol with a program*.
3. In the list, select the `.xdp` file type and select *Change program...*.
4. Select *Browse* and then the `FormDesigner.exe` file in the folder where you installed the Adobe LiveCycle Designer program files. Click *Open*.
5. Make sure that the `Adobe LiveCycle Designer ES` entry and the *Always use the selected program to open this kind of file* checkbox are selected. Click *OK*.
   When you now double-click an `.xdp` file to open it, the file opens in Adobe LiveCycle Designer.

8.10.3 Create a Print Form

You can create a print form to output the data from a business object. You use Adobe LiveCycle Designer to define the data you want to include on the form and design the layout of the form.

Prerequisites

- Your solution contains an activated business object with at least one element.
- You have installed Adobe LiveCycle Designer.
- You have associated the `.xdp` file type with Adobe LiveCycle Designer.
  For more information, see Associating an .xdp File with Adobe LiveCycle Designer [page 508].

Procedure

1. In the *Solution Explorer*, right-click the business object and select *Create Print Forms*.
   The *Form Wizard* opens.
2. In the *Define Form Properties* step, under *Basic Information*, you can select a different business object.

3. Under *Form Template Header*, do the following:
   - Optional: Change the name of the form template header.
   - Enter a description for the form template header.
4. Under *Form Template Group*, select a form template group (if available) or click *New* to create a new group.
5. If you selected *New*, in the *Form Template Editor* window, accept the default name or change the name of the group in the *Name* field, click *Save*, and then click *Next*.
6. In the *Define Form Data Structure* step, the data structure of the business object is displayed. Select the data elements you want to include on the form, and then click *Next*.
7. In the *Define Properties of Default Variant* step, enter a *Comment* to describe this form and select a *Language*.
   You can select one of the languages in which the SAP cloud solution is available.
8. Click *Next*.
   A summary of the settings you entered for the form is displayed.
9. In the *Review Form Settings* step, click *Finish*.
   The form is opened in Adobe LiveCycle Designer.

> → Tip
>
> If Adobe LiveCycle Designer does not open and an error message is displayed instead, make sure that you have associated the `.xdp` file type with Adobe LiveCycle Designer. For more information, see Associating an .xdp File with Adobe LiveCycle Designer [page 508].

10. To create your form, drag data elements from the *Data View* on the left to the *Design View* on the right.

> i Note
>
> You may need to enlarge the left pane to see the *Data View* tab.

11. Once you have designed the form, save your work and close Adobe LiveCycle Designer.
    For more information about using Adobe LiveCycle Designer, see the Adobe product documentation.
12. In the *Solution Explorer*, right-click the form group file (`.ftgd`,) and select *Check In Content*. Once the check-in is complete, right-click the file and select *Activate*.

Results

- A print form is associated with the business object. You now need to make the form accessible to users from a floorplan. For more information about how to enable a preview function that allows users to preview the form as a PDF and provides options to save or print the form, see Create a Preview Button for a Print Form [page 734].
- You can now test your print form. For more information, see Test a Print Form [page 510].

Related Information

Print Forms Quick Guide [page 506]

8.10.4 Test a Print Form

You can test the print form in the preview after deploying the business configuration content of your solution.

Prerequisites

You must be assigned the `Business User` role and have access rights for the *Business Configuration* work center. For more information, see User Setup Quick Guide [page 33].

Procedure

1. **Perform Business Configuration Deployment**
   There are two ways to deploy your business configuration content: Either you scope the BAC elements defined in your solution in the *Business Configuration* work center of the SAP cloud solution or you select the *Deploy Business Configuration* function in the context menu of the your solution.
   - If you have created BAC elements in your solution and assigned your solution content to a business option, you can perform scoping manually in the *Business Configuration* work center of the SAP cloud solution.
     For more information, see "Configuring your SAP Solution" in the documentation in your SAP cloud solution.
   - Alternatively, you can trigger the deployment of business configuration in the *Solution Explorer* by right-clicking the solution and selecting *Deploy Business Configuration*.

     > **i Note**
     >
     > When you use the *Deploy Business Configuration* function, the studio deploys all business configuration content of your solution. However, you cannot test partial activation of business configuration content in the development environment because the *Deploy Business Configuration* function deploys all business configuration content independently of any business options you have created.

2. **Test the Print Form**
   1. In the *Solution Explorer*, right-click the floorplan and click *Preview Screen*.

      > **i Note**
      >
      > If this is the first time you have selected this option, you may be prompted to enter a user name and password.

      A preview of the screen is displayed.
   2. On the screen, enter some values in the fields and click *Save*.
   3. Click *Preview*.

      > **i Note**
      >
      > If more than one template exists, a list of available templates is displayed in the *Preview* window. Select a template and click *Open*.

A preview of the selected print form is displayed as a PDF.

8.11 Analytics

8.11.1 Analytics Quick Guide

Analytics in the cloud solution allows customers to support and monitor business processes and thus enables them to make informed decisions. The standard system provides predefined analytics content that is presented in reports. The data from key figures and characteristics in a data source or a key figure group serves as the basis of these reports. Administrators can design reports that they can make available to other users. Users can then access these reports from the *Reports* work center view of the corresponding work centers or from the *Overview* work center views.

In the studio, you can add data sources to a solution that administrators for Analytics can use for reporting in the cloud solution. You can also create browser-based reports in the studio and use the administrator tools for Analytics in the cloud solution to define the report layout and content.

Business and Technical Background

Defining Analytics Content

In the studio, you can define analytics content for custom business objects, for SAP business objects, or for a combination of both. In the SAP cloud solution, customers use this content for reporting. Reports show the data and values derived from key figures and characteristics in data sources, key figure structures, and key figure grids.

For more information, see here [page 515].

Data Sources

A data source is an object that provides a multidimensional, analytical view of business data. Data sources are the basis of analytical report content and are associated with a specific access context or can be unrestricted.

For more information, see here [page 517].

Combined Data Sources

A combined data source combines data from two or more data sources that share common or similar characteristics.

For more information, see here [page 519].

Joined Data Sources

A joined data source joins data from two or more data sources that share common characteristics.

For more information, see here [page 520].

Key Figures

Fields according to which values are selected. Key figures are numeric values that have a unit of measure or currency assigned. Examples include Invoice Net Value and Purchase Order Quantity.

Transformation Rules

You can use transformation rules to calculate fields from business object fields, for example, for date and date/time conversions. The transformed (calculated) fields are added to the data source and can be used in reports.

For more information, see here [page 523].

Aggregation

To enable the calculation of key figures, the system can aggregate key figure values from a data source based on defined aggregation behavior. The system aggregates key figure values for multiple characteristics. The system can also aggregate key figure values for a specified characteristic; this is referred to as exception aggregation.

For more information, see here [page 525].

Maintenance of Analytics Content

A customer-specific solution is in maintenance mode, if it has been assembled and downloaded; the solution status then is *Assembled*. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.

For information about creating a patch, see Patches [page 178].

For information about the change and delete restrictions for data sources in a solution that is in maintenance mode, see Maintenance of Analytics Content [page 526].

Tasks

Create a Data Source

In the studio, you can create a data source based on an SAP business object or a custom business object. A data source includes specific data such as key figures and characteristics that administrators for Analytics can use for reporting in the SAP cloud solution.

For more information, see here [page 527].

Change a Data Source

1. In the studio, open the solution that contains the data source you want to change.
2. In the *Solution Explorer*, double-click your data source.
   The data source opens in a document window and the information you defined using the *Data Source Wizard* is displayed on different tabs.
3. Make your changes and then click *Save*.
   You cannot change the entries in the mandatory fields (*Namespace*, *Business Object*, *Anchor Node*, and *Data Source Name*) on the *Header Information* tab.
   The system saves your changes.

> **i Note**
>
> For information about the change and delete restrictions for data sources in a solution that is in maintenance mode, see Maintenance of Analytics Content [page 526].

Delete a Data Source

1. In the studio, open the solution that contains the data source you want to delete.
2. In the *Solution Explorer*, right-click the data source and select *Delete*.
   If other project items depend on a data source, for example, if a data source is used in a report or a joined data source, you cannot delete this data source. To delete the data source, you must first delete any dependent project items.

> **i Note**
>
> For information about the change and delete restrictions for data sources in a solution that is in maintenance mode, see Maintenance of Analytics Content [page 526].

Create a Combined Data Source

In the SAP cloud solution, you can use the administrator tools for Analytics to create a combined data source that combines data from multiple data sources.

For more information, see here [page 530].

Create a Joined Data Source

In the SAP cloud solution, you can use the administrator tools for Analytics to create a joined data source that joins data from multiple data sources that share common characteristics.

For more information, see here [page 532].

Create a Key Figure

In the SAP cloud solution, you can use the administrator tools for Analytics to create a key figure based on an SAP data source or based on a data source you have created (custom data source). This key figure can then be used in a report. There are two types of key figures: A **calculated key figure** is determined using calculation rules or formulas and a **restricted key figure** is restricted to a specified characteristic value.

For more information, see here [page 535].

Create a Report

In the studio, you can create reports by using data sources that you have created. You can then define the report and the report layout in the SAP cloud solution by using the administrator tools for Analytics.

For more information, see here [page 534].

Manage Reports

1. In the *Solution Explorer*, right-click your data source and select *Manage Reports*.
   The logon screen of the cloud solution opens in your browser.
2. Enter the user name and password that you entered when logging on to the repository.
   A report list opens and displays the reports assigned to your data source.
3. Optional: Change the content and layout of a report by doing the following:
   - To change the report definition, click *Edit With* and then select *Report Wizard*.
   - To change the report layout, click *Edit With* and then select *Web Browser*.

Change a Report Created in an Earlier Release

When you maintain a report that you created in an earlier release of your solution, you must ensure that the report has the same technical ID in all later releases to ensure that multiple versions of the report do no exist with different IDs and that all analytics content created by customers, for example, the field names of a data source, still has valid references. To change the technical ID, you must find out the technical ID that the system assigned to the initial version of your report and manually assign this ID to the report in the current release. To do this, do the following:

1. Open the initial version of your report in the development environment of the cloud solution and find the technical ID of the report by doing the following:
    1. In the *Solution Explorer*, right-click the data source on which your report is based and select *Manage Reports*.
    The logon screen of the cloud solution opens in your browser.
    2. Enter the user name and password that you entered when logging on to the repository with your Developer user role.
    The *Design Reports* view opens and displays the reports assigned to the data source including your report.
    3. Select your report, click *Edit With* and then select *Report Wizard*.
    The *Report Wizard* opens and displays the *Define Report and Select Data Source* step.
    4. At the top right of the screen, click *Personalize* and select *This Screen*.
    The *Personalize* panel opens.
    5. Under *Sections*, select *Report* and make sure that the *Visible* option is selected.
    6. Under *Fields*, find the *Technical ID* field and select the *Visible* option.
    7. Save your changes and close the *Personalize* panel.
    The screen now displays the technical ID of the report. Make a note of the ID.
2. Open your report in the current release of your solution and repeat the steps described above to personalize the screen of the *Define Report and Select Data Source* step.
3. Overwrite the ID displayed in the *Technical ID* field with the ID of the initial version of your report and save your changes by clicking *Finish*.
The technical ID of the report is now the same in the initial release and in the current release of your solution.

8.11.2 Business and Technical Background

8.11.2.1 Defining Analytics Content

Overview

In the studio, you can define analytics content for custom business objects, for SAP business objects, or for a combination of both. In the SAP cloud solution, customers use this content for reporting. Reports show the data and values derived from key figures and characteristics in data sources, key figure structures, and key figure grids.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Prerequisites

- You have created and opened a solution in the studio. For more information, see Create a Customer-Specific Solution [page 177].
- You understand how administrators design reports in the SAP cloud solution.
- You are assigned the Developer user role and have access rights for the *Business Analytics* work center. For more information, see User Setup Quick Guide [page 33].

Process Flow

1. You create a data source.
   - In the studio, you add a data source to your solution and define the data source. For more information, see Create a Data Source [page 527].
     You must activate the data source to make it available in the SAP cloud solution so that you can use it for your report content.
   - You use the administrator tools for Analytics in the SAP cloud solution to create a combined data source or a joined data source.
     For more information, see Create a Combined Data Source [page 530] and Create a Joined Data Source [page 532].
2. In the studio, you add a report to your solution.
   You must enable the admin mode in the studio so that the system saves all reports and report views you create in your solution.
   For more information, see Create a Report [page 534].
3. in the SAP cloud solution, you define your report and the report layout in the *Design Reports* view of the *Business Analytics* work center.
   For more information, see the relevant documentation of the SAP cloud solution.
4. Optional: In the studio, you create a work center and a work center view to which you want your report to be assigned; then, in the UI designer, you:
   1. Add the work center view to the work center.
   2. Assign a stable anchor to the work center view to which you want your report to be assigned.
      For more information, see Define an Anchor for a Floorplan [page 753] and Anchors [page 759].
      You must assign this anchor so that administrators can assign reports to this work center view in the SAP cloud solution.

3. Add the *List* and *Gallery* standard reports list views to your work center. These views list all reports that are assigned to the views of the work center.
   For more information, see Add a Report List View to a Work Center [page 752].

5. In a customer's production system, an Analytics administrator assigns the report to the work center view to make it available for business users.

> **i Note**
>
> Only administrators for the SAP cloud solution working in a customer's production system can assign a report that you have created to a work center view. To assign a report to a work center view, the solution that contains the report must therefore be implemented in the customer's production system.
>
> We recommend that you provide information about the work center views to which customers need to assign reports in your Solution Documentation. For more information, see Documentation Types in Cloud Solutions from SAP [page 822].

Related Information

Analytics Quick Guide [page 511]

8.11.2.2 Data Sources

Overview

A data source is an object that provides a multidimensional, analytical view of business data. Data sources are the basis of analytical report content and are associated with a specific access context or can be unrestricted.

Data sources contain the following objects:

- Key figures
  Fields according to which values are selected. Key figures are numeric values that have a unit of measure or currency assigned. Examples include Invoice Net Value and Purchase Order Quantity.
- Characteristics
  Fields according to which values are selected. Characteristics are alphanumeric, numeric, or text values. Examples include Product ID, Supplier, and Purchase Order Status. Variables are often associated with specified characteristics. Variables restrict characteristics to one or more specified value selections.

Availability

> **⚠ Caution**
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud

solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.

For details, see Feature Matrix [page 27].

Working with Data Sources in the Studio

In the studio, you can create a data source based on an SAP business object or a custom business object. A data source includes specific data such as key figures and characteristics that administrators for Analytics can use for reporting in the SAP cloud solution.

When you define a data source, you select the relevant fields from your business objects, from SAP business objects, or from a combination of both. You can join the fields of multiple business objects by using associations. The system supports [1..1] associations with a foreign-key relationship.

Optionally, you can:

- Define transformation rules.
  For more information, see Transformation Rules [page 523].
- Reference other data sources.
- Define aggregation rules to summarize key figures (*Summation*, *Minimum*, *Maximum*, *Count*) in a report using characteristics.
  For more information, see Aggregation [page 525].

The studio supports read access across deployment units in data sources. You can use cross-deployment-unit associations to access the data in business objects that are not assigned to the default deployment unit of your solution. For more information, see Syntax for Business Object Definitions [page 191] under *Deployment Unit*.

Working with Data Sources Using the Administrator Tools of the SAP Solution

You can join or combine data from different data sources that share common characteristics and contain semantically different key figures. For more information, see Combined Data Sources [page 519] and Joined Data Sources [page 520].

Related Information

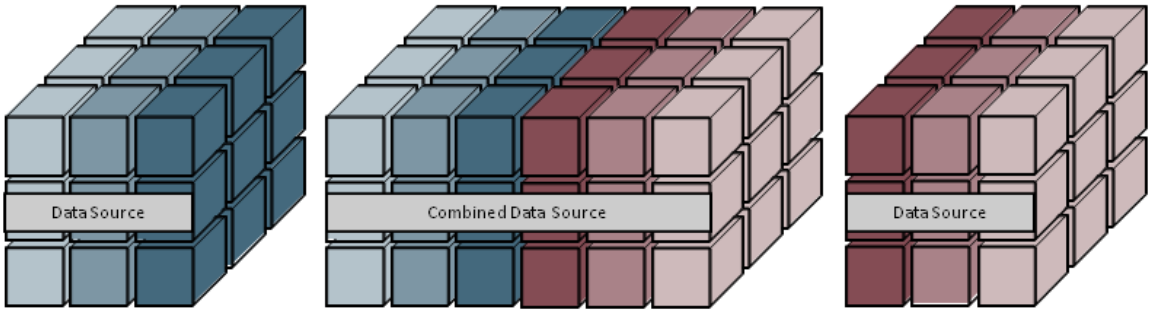
Analytics Quick Guide [page 511]
Create a Data Source [page 527]
Create a Combined Data Source [page 530]
Create a Joined Data Source [page 532]

8.11.2.3 Combined Data Sources

Overview

A data source is an object that provides a multidimensional, analytical view of business data. Data sources are the basis of analytical report content and are associated with a specific access context or can be unrestricted.

A combined data source combines data from two or more data sources that share common or similar characteristics.



Depiction of a Combined Data Source

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Use

In the SAP cloud solution, you can use the administrator tools for Analytics to create a combined data source that combines data from multiple data sources.

When you build a report using a combined data source, all characteristic values are displayed along with any key figure values.

You usually combine data sources that have the semantically similar characteristics. However, you can also combine data sources when the characteristics you require for a report are not available in one data source.

The following example shows a result set of a combined data source.

Data Source

Product ID	Supplier	Purchase Order ID
MCD-0001	1000010	1
MCD-0002	1000011	2
MCD-0022	1000012	3
MCD-0024	1000013	4
MCD-0005	1000014	5

Data Source

Product ID	Product Category	Supplier
MCD-0001	Service	1000010
MCD-0002	Service	1000011
MCD-0022	IT Equipment	1000012
MCD-0024	IT Equipment	1000013
MCD-0005	Service	1000014
MCD-0046	Office Supplies	1000215
MCD-0049	Office Supplies	1000215

Combined Data Source

Product ID	Supplier	Product Category	Purchase Order ID
MCD-0001	1000010	#	1
MCD-0001	1000010	Service	#
MCD-0002	1000011	#	2
MCD-0002	1000011	Service	#
MCD-0005	1000014	#	5
MCD-0005	1000013	Service	#
MCD-0022	1000012	#	3
MCD-0022	1000012	IT Equipment	#
MCD-0024	1000013	#	4
MCD-0024	1000012	IT Equipment	#
MCD-0046	1000215	Office Supplies	#
MCD-0049	1000215	Office Supplies	#

Example of a Result Set of a Combined Data Source

Related Information

Analytics Quick Guide [page 511]
Create a Combined Data Source [page 530]

8.11.2.4 Joined Data Sources

Overview

A data source is an object that provides a multidimensional, analytical view of business data. Data sources are the basis of analytical report content and are associated with a specific access context or can be unrestricted.

A joined data source joins data from two or more data sources that share common characteristics.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.

For details, see Feature Matrix [page 27].

Use

In the SAP cloud solution, you can use the administrator tools for Analytics to create a joined data source that joins data from multiple data sources that share common characteristics.

When you build a report using a joined data source, the characteristic values that match in the joined characteristics in the data sources are displayed along with the values that are in the same row.

Joined data sources can have more than one join. A join is created when a characteristic is available in a data source that is an anchor. By default, the first data source you select is the anchor. The more joins there are in a joined data source, the more restricted the data is that is available for reports.

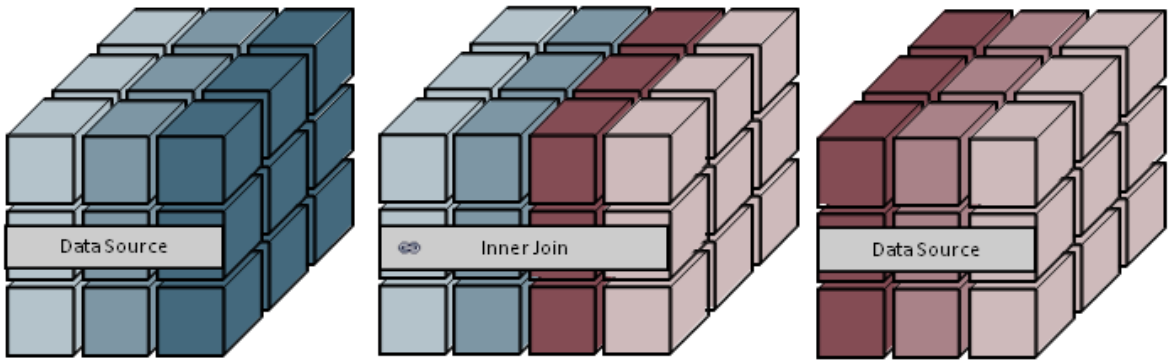
Join Types

There are different types of joins between data sources. You select a join type depending on your business requirements.

> **i Note**
>
> The anchor is displayed for both join types in the system but only affects the data in a joined data source that uses a left outer join.

Inner Join

An inner join selects only those characteristic values from the data sources to be joined that match in the joined data source field along with any key figure values. Characteristic values in the joined data source fields that do not appear in both data sources are excluded from the joined result set.



Depiction of an Inner Join

The following example shows a result set of a joined data source that uses an inner join.

Data Source

Customer ID	Last Name	First Name	City
1	Adams	Oliver	Chicago
2	Menson	Bob	Miami
3	Jacob	Kate	Boston

Data Source

Customer ID	Order Number	Order ID
3	55234	1
3	55233	2
1	55199	3
1	54987	4
7	55340	5

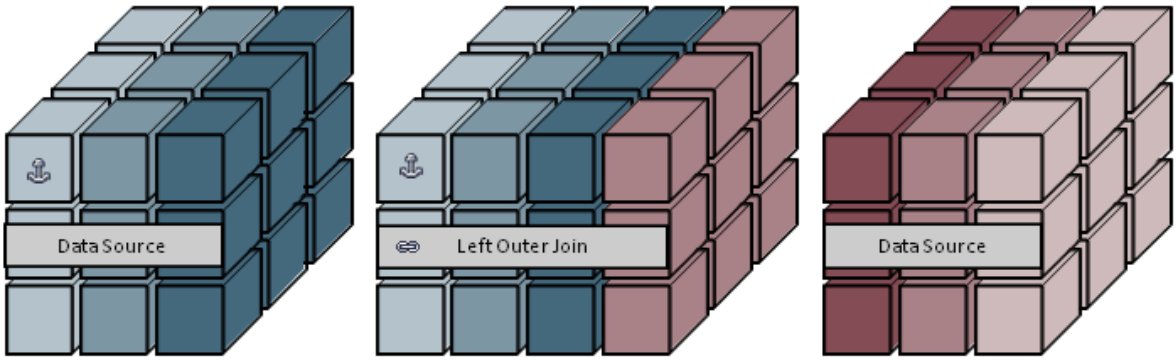
Inner Join

Customer ID	Last Name	First Name	City	Order Number
1	Adams	Oliver	Chicago	55199
1	Adams	Oliver	Chicago	54987
3	Jacob	Kate	Boston	55234
3	Jacob	Kate	Boston	55233

Example of a Result Set of a Joined Data Source That Uses an Inner Join

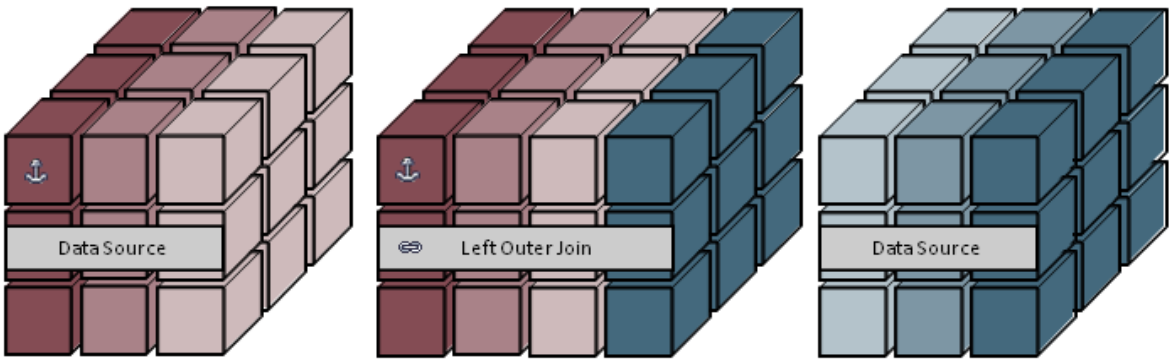
Left Outer Join

A left outer join selects all characteristics values from the data source that is selected as the anchor and those characteristic values that match in the data sources that are added along with any key figure values. By default, the first data source that you select is the anchor.



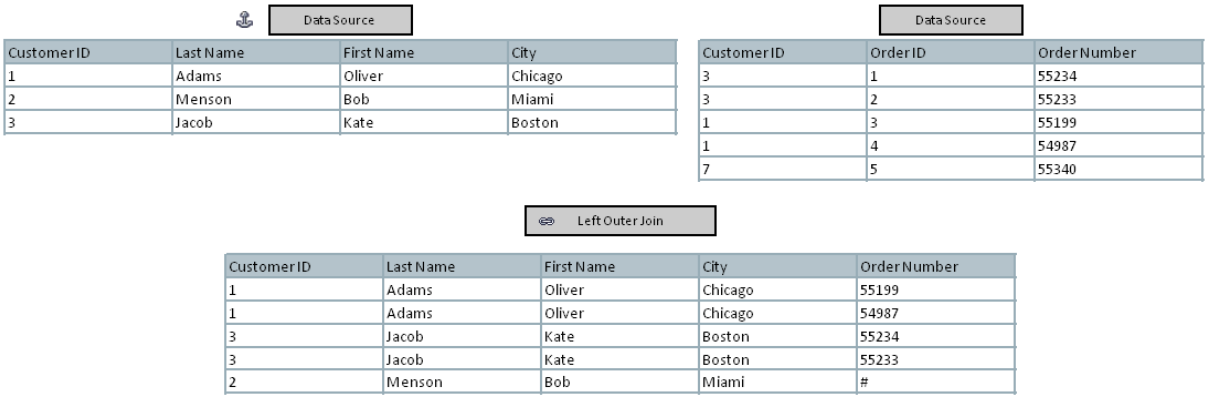
Depiction of a Left Outer Join

> ### i Note
>
> Setting the anchor to a different data source affects the values available because the joined data source fields may also change.



Depiction of an Anchor Change

The following example shows a result set of a joined data source that uses a left outer join.



Example of a Result Set of a Joined Data Source That Uses a Left Outer Join

Related Information

Analytics Quick Guide [page 511]
Create a Joined Data Source [page 532]

8.11.2.5 Transformation Rules

Overview

You can use transformation rules to calculate fields from business object fields, for example, for date and date/time conversions. The transformed (calculated) fields are added to the data source and can be used in reports.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Use

Business objects often contain date and date/time information, for example, for the release of an order. In reports, you normally want to aggregate data in time buckets, for example, to view the orders per month. For

this purpose, you could define a `DateToMonth` or `DatetimeToMonth` transformation rule. As a result, the system adds a new field to the data source that contains only the month of the release `datetime` field. In a report, you can then easily aggregate the order data by using the month field.

You select transformation rules in the *Data Source Wizard*. For information about how to create a data source using the *Data Source Wizard*, see Create a Data Source [page 527].

Available Transformation Rules

Transformation rules are only available for fields of the data types `GLOBAL_DateTime`, `Date`, and the `Name`, `Description`, and `Text` data types, which represent character strings.

Transformation Rules for Date/Time Fields (GLOBAL_DateTime Data Type)

Rule	Description
`DatetimeToTime`	Selects and retrieves only the time from a `datetime` field.
`DatetimeToDay`	Selects and retrieves only the day from a `datetime` field.
`DatetimeToDate`	Selects and retrieves only the date from a `datetime` field.
`DatetimeToCalendarweek`	Calculates the calendar week of a `datetime` field.
`DatetimeToMonth`	Selects and retrieves only the month from a `datetime` field.
`DatetimeToQuarter`	Calculates the quarter of a `datetime` field.
`DatetimeToYear`	Selects and retrieves only the year from a `datetime` field.
`DatetimeToYearmonth`	Selects and retrieves only the year and month from a `datetime` field.
`DatetimeToYearquarter`	Calculates the year and quarter of a `datetime` field.
`DatetimesToDuration`	Calculates the number of days between two `datetime` fields. *

Transformation Rules for Date fields

Rule	Description
`DateToMonth`	Selects and retrieves only the month from a `date` field.
`DateToYearmonth`	Selects and retrieves only the year and month from a `date` field.
`DatesToDuration`	Calculates the number of days between two `date` fields. *

Transformation Rules for String Fields

Rule	Description
`Concatenate`	Concatenates two string fields. *

> **i** Note
>
> * To use this transformation rule, you must select a second field.

8.11.2.6 Aggregation

Overview

To enable the calculation of key figures, the system can aggregate key figure values from a data source based on defined aggregation behavior. The system aggregates key figure values for multiple characteristics. The system can also aggregate key figure values for a specified characteristic; this is referred to as exception aggregation.

Availability

> **⚠** Caution
>
> This documentation covers the SAP Cloud Applications Studio ("studio") as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Features

The studio supports the definition of standard aggregation. You can define exception aggregation behavior in the SAP cloud solution using the administrator tools for Analytics.

Standard Aggregation

The standard aggregation behaviors you can define in the studio are summation, minimum, maximum, and count. For example, you can define minimum and maximum aggregation for date key figures. You define aggregation behavior in the studio using the *Data Source Wizard*. For information, see Create a Data Source [page 527].

Exception Aggregation

You define exception aggregation behavior for values of a calculated key figure. The deviation to the standard aggregation behavior is only valid in combination with a specified characteristic. You define exception aggregation behavior in the SAP cloud solution using the *Report Wizard*.

Related Information

Data Sources [page 517]

8.11.2.7 Maintenance of Analytics Content

Overview

A customer-specific solution is in maintenance mode, if it has been assembled and downloaded; the solution status then is *Assembled*. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio ("studio") as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Change and Delete Restrictions

The following table gives an overview of the change and delete restrictions for analytics content in a solution that is in maintenance mode. The system performs the change and delete restrictions checks when you save the data source.

Maintenance of Data Sources

Action	Allowed
Delete a data source	No
Delete data source fields	No
Add or delete new data source fields	Yes
Change or delete transformations	No
Add or delete new transformations	Yes
Change the *Display Name* and *Referenced Data Source* fields of data source fields	No
Change or delete aggregations	No
Add or delete new aggregations	Yes
Create (and delete) a new data source	Yes
Delete a reference to a joined data source created by an administrator in the cloud solution	Yes
Delete a reference to a combined data source created by an administrator in the cloud solution	Yes
Delete a reference to a key figure created by an administrator in the cloud solution	Yes

Maintenance of Reports

Action	Allowed
Delete a report	Yes

Related Information

Analytics Quick Guide [page 511]
Lifecycle Management of Customer-Specific Solutions [page 143]
Maintenance Mode [page 185]

8.11.3 Tasks

8.11.3.1 Create a Data Source

In the studio, you can create a data source based on an SAP business object or a custom business object. A data source includes specific data such as key figures and characteristics that administrators for Analytics can use for reporting in the SAP cloud solution.

Prerequisites

- You have created and opened a solution in the studio. For more information, see Create a Customer-Specific Solution [page 177].
- To create a data source based on a custom business object, you must create and activate your custom business object.
  For more information, see Business Objects Quick Guide [page 398].
- If you want to allow referencing of your data source by another data source and enable navigation from a report to another floorplan that is not a standard UI, you must define this UI. For information, see Generate Screens for a Business Object [page 429].

Procedure

1. In the *Solution Explorer*, do the following:

Custom Business Object	SAP Business Object
Right-click your business object and select *Create Data Source*. The *Data Source Wizard* opens.	1. Right-click your solution and select *Add* > *New Item*. The *Add New Item* dialog opens. 2. Select *SAP* > *Analytics* > *Data Source* and click *Add*. The *Data Source Wizard* opens.

2. In the *Create a Data Source* step, define your data source by doing the following:
   1. Select a namespace.
      By default, the namespace for your solution is selected; however, to create a data source based on an SAP business object, you can select an SAP namespace.
   2. Select a business object.
   3. Select an anchor node to link to the data source.

      → Tip

      - If a [1,n] association is defined for a field, the field is not displayed in the business object hierarchy. Therefore, we recommend that you select the most detailed node as the anchor node. For more information about associations, see Syntax for Business Object Definitions [page 191] under *Association*.
      - You can also select nodes of dependent objects, for example, `TextCollection.Root`.
   4. Enter the data source name.
   5. Optional: Enter a description of the data source.
   6. Optional: To allow access restrictions for data in reports based on this data source, select *Enable Access Control*.
      The access restrictions of the anchor business object are inherited; therefore customers can only assign reports based on this data source to work center views with the same access control.
   7. Optional: To allow referencing by another data source and allow the fields of your data source to be used in reports based on the referencing data source, select *Allow Referencing*.
      - If you want to allow referencing, under *Reference Details*, select the key field.

- Optional: Under *Object Based Navigation*, specify a target UI.
  Users will be able to navigate from the report to this floorplan.

8. Optional: To enable read access logging for sensitive personal data, select the *Log Sensitive Personal Data* checkbox.
9. Click *Next*.

3. In the *Select Data Source Fields* step, select the fields you want to include in the data source.
   You can include fields from other business objects by using associations. The studio supports read access across deployment units in data sources. You can use cross-deployment-unit associations to access the data in business objects that are not assigned to the default deployment unit of your solution. For more information, see Deployment (Business Object) [page 205].

   > **i Note**
   >
   > - If your datatype requires a context, for example, the region needs the country as a context, you usually maintain this context as compounding. However maintaining compounding is not possible in the studio.
   > - Not all fields and associations modeled in the business object are supported. Only supported fields and associations are displayed.

   Click *Next*.
4. Optional: In the *Define Transformation Rules* step, define transformation rules for fields of data type text (such as name and description), date, or `GLOBAL_DataTime` by doing the following:
   1. In the *Business Object Element* area, select a field and on the *Transformation* tab, select a transformation rule.
   2. Define a field name for the transformation field and then press **ENTER**.
   3. If the transformation rule allows the selection of a second parameter, next to the *Parameter* field, click ....
      The *Select Additional Transformation Rule Parameter* dialog box opens.
   4. In the *Select Additional Transformation Rule Parameter* dialog box, select a second parameter by selecting a row and clicking the first cell in the row.
      For information about the available transformation rules, see Transformation Rules [page 523].
   5. Click *Next*.
5. Optional: In the *Define Referenced Data Sources* step, modify the parameters by doing the following:
   1. Display and define the names for the selected fields that are characteristics.
   2. Assign an existing data source to each parameter.

      > **i Note**
      >
      > If necessary, you can change the display name for each parameter.
      >
      > You can assign an existing data source if the fields of an SAP business object are already included in an SAP data source.

   3. Click *Next*.
6. Optional: In the *Define Aggregation Behavior* step, define the aggregation behavior for business object elements that are key figures.
   For more information about aggregation behavior, see Aggregation [page 525].

> **Note**
>
> The default aggregation of every key figure is *Summation*. You can change the default or add new aggregations.

7. Optional: In the *Select Sensitive Personal Data Fields* step include sensitive personal data fields for read access logging by selecting the *Create Log* checkbox against the field. You must then include a reference for the sensitive personal data field.

   > **Note**
   >
   > If you select a field that has transformation rules, you can include only the parent field as the reference.

8. In the *Review* step, check that the information you have entered is correct and then click *Finish*.

   > **Note**
   >
   > If you have defined aggregations, check that the selected aggregations are displayed correctly.

   A new item with the file extension `.ds` ( ) appears in the *Solution Explorer*.

9. In the *Solution Explorer*, right-click the data source you have created and select *Activate*.

Result

The data source is available in SAP's cloud solution. You can check the result by logging on to SAP's cloud solution. In the *Business Analytics* work center, navigate to the *Design Data Source* view and search for your data source.

In the studio, you can now create a report that uses your data source. For more information, see Create a Report [page 534].

You define the report and the report layout in SAP's cloud solution in the *Design Reports* view of the *Business Analytics* work center. For more information, see the relevant documentation of SAP's cloud solution.

8.11.3.2 Create a Combined Data Source

In the SAP cloud solution, you can use the administrator tools for Analytics to create a combined data source that combines data from multiple data sources.

You start the process in the studio and then create the combined data source on the *New Combined Data Source* screen of the *Business Analytics* work center. For information about combined data sources, see Combined Data Sources [page 519] and the relevant documentation of the cloud solution.

Prerequisites

- You are assigned the Developer user role and have access rights for the *Business Analytics* work center. For more information, see User Setup Quick Guide [page 33].
- You have switched on the admin mode so that the system saves the analytics content you create in your solution. To do this, select *Administration* > *Switch On/Off Admin Mode in SAP Solution*.

Procedure

1. In the *Solution Explorer*, right-click your solution and select *Add* > *New Item*
   The *Add New Item* dialog opens.
2. Select *SAP* > *Analytics* > *Combined Data Source*.
3. Enter a name and click *Add*.
4. The SAP cloud solution opens. If you are not logged in already, the logon screen opens. Log on with your developer user.
5. The *New Combined Data Source* screen opens. By default, the combined data source contains the *Counter* key figure. The *Counter* key figure counts the number of instances of a specified object.
6. On the screen, click *Add Data Source*.
7. On the *Add Data Source* dialog screen, select a data source from the dropdown list.
   The screen displays the characteristics and key figures available in the selected data source. The initial selections serve as a basis for the combined data source. The characteristics and key figures are then displayed in the *Combined Data Source Fields* column.

   > **i Note**
   >
   > You can only combine data sources that have compatible access contexts. For example, you can add a data source if the data source and the combined data source have a common access context. You can add any data source that has an unrestricted access context to a combined data source.

8. Select one or more additional data sources and add the data sources to the combined data source.
   The system checks whether the characteristics and key figures in the additional data sources are also available in the data source that you selected as a basis for the combined data source.

   > **i Note**
   >
   > If a characteristic or key figure is available in the combined data source, the system positions the characteristic or key figure in the same row of each data source.
   >
   > If a characteristic or key figure is not available in the combined data source, the system adds the characteristic or key figure to the combined data source.

   Other Functions

   - *Remove Data Source*
     The *Remove Data Source* dialog screen opens and displays the data sources you have added. To remove data sources, click the relevant checkboxes.
   - *Add Row*
     Adds a row to which you can add a characteristic or key figure. The system then adds the characteristic or key figure to the relevant field type group.

- *Remove Row*
  Deletes the selected row.

  > **Note**
  >
  > The selected characteristic or key figure is only removed from the combined data source.

9. Save your entries.
10. In the studio, in the *Solution Explorer*, update your solution by clicking *Refresh Solution* ( ).

    A `.cds` file ( ) appears under your solution.

Result

The combined data source is available for reporting. You can create reports using data from the combined data source. For information about creating reports, see Create a Report [page 534].

You can delete the combined data source when your solution is in maintenance mode. For more information, see Maintenance Mode [page 185].

8.11.3.3 Create a Joined Data Source

In the SAP cloud solution, you can use the administrator tools for Analytics to create a joined data source that joins data from multiple data sources that share common characteristics.

You start the process in the studio and then create a joined data source on the *New Joined Data Source* screen of the *Business Analytics* work center. For information about joined data sources, see Joined Data Sources [page 520] and the relevant documentation of the cloud solution.

Prerequisites

- You are assigned the Developer user role and have access rights for the *Business Analytics* work center. For more information, see User Setup Quick Guide [page 33].
- You have switched on the admin mode so that the system saves the analytics content you create in your solution. To do this, select *Administration* > *Switch On/Off Admin Mode in SAP Solution*.

Procedure

1. In the *Solution Explorer*, right-click your solution and select *Add* > *New Item*
   The *Add New Item* dialog opens.
2. Select *SAP* > *Analytics* > *Joined Data Source*.

3. Enter a name and click *Add*.
4. The SAP cloud solution opens. If you are not logged in already, the logon screen opens. Log on with your developer user.
5. The *New Joined Data Source* screen opens.
   In the *Join Type* field, the *Inner Join* option is preselected. You can also select the *Outer Left Join* option.
   For more information, see Joined Data Sources [page 520]

   > **i Note**
   >
   > By default, the joined data source contains the *Counter* key figure. The *Counter* key figure counts the number of instances of a specified object.

6. On the *New Joined Data Source* screen, click *Add Data Source*.

   > **i Note**
   >
   > You cannot join more than five data sources.

7. On the *Add Data Source* dialog screen, select a data source from the dropdown list.
   The screen displays the characteristics and key figures available in the selected data source. The initial selections serve as a basis for the joined data source. The characteristics and key figures are then displayed in the *Joined Data Source Fields* column.
   By default, the first data source you select is the anchor.
8. Select one or more additional data sources and add the data sources to the joined data source.
   The system checks whether the characteristics and key figures in the additional data sources are also available in the data source that you selected as a basis for the joined data source.

   > **i Note**
   >
   > If a characteristic or key figure is available in the joined data source, the system positions the characteristic or key figure in the same row of each data source.
   >
   > If a characteristic or key figure is not available in the joined data source, the system adds the characteristic or key figure to the joined data source.
   >
   > If a characteristic is available in a data source that is an anchor, the system creates a join. The more joins there are in a joined data source, the more restricted the data is that is available for reports.

   Other Functions

   - *Remove Data Source*
     The *Remove Data Source* dialog screen opens and displays the data sources you have added. To remove data sources, click the relevant checkboxes.
   - *Add Row*
     Adds a row to which you can add a characteristic or key figure. The system then adds the characteristic or key figure to the relevant field type group.
   - *Remove Row*
     Deletes the selected row.

     > **i Note**
     >
     > The selected characteristic or key figure is only removed from the joined data source.

   - *Check*
     Checks whether the joined data source is consistent.

- *Set Anchor*
  Sets the anchor to the selected data source.
  The data source that is the anchor is the basis for any joins. Depending on the type of join, changing the anchor to a different data source can change the joined data source fields.
- *Set Access Context*
  Sets the access context to the access context of the selected data source.

9. Save your entries.
10. In the studio, in the *Solution Explorer*, update your solution by clicking *Refresh Solution* ( ).

    A `.jds` file ( ) appears under your solution.

Result

The joined data source is available for reporting. You can create reports using data from the joined data source. For information about creating reports, see Create a Report [page 534].

You can delete the joined data source when your solution is in maintenance mode. For more information, see Maintenance Mode [page 185].

8.11.3.4 Create a Report

In the studio, you can create reports by using data sources that you have created. You can then define the report and the report layout in the SAP cloud solution by using the administrator tools for Analytics.

You can open a report that you created in the studio in the *Design Reports* view in the development environment with the Developer user role. You cannot open reports in the development environment with the Business User role.

Prerequisites

- You have switched on the admin mode so that the system saves the analytics content you create in your solution. To do this, select *Administration* > *Switch On/Off Admin Mode in SAP Solution* .
- You are assigned the Developer user role and have access rights for the *Business Analytics* work center. For more information, see User Setup Quick Guide [page 33].
- You have created and activated a data source. For more information, see:
  - Create a Data Source [page 527]
  - Create a Combined Data Source [page 530]
  - Create a Joined Data Source [page 532]

Procedure

1. In the *Solution Explorer*, right-click your data source and select *Create Report*.
   The logon screen of the cloud solution opens in your browser.
2. Enter the user name and password that you entered when logging on to the repository.
   The *Report Wizard* opens.
3. Use the *Report Wizard* to define the report characteristics, key figures, variables, and properties; then define the report layout.
   For more information, see "Working with Reports Using the Report Wizard" in the documentation of the SAP cloud solution.
   The system displays the name of your solution. The report ID starts with the default namespace.
4. In the studio, in the *Solution Explorer*, update your solution by clicking *Refresh Solution* ( ).

   A `.report` file ( ) appears under your solution.

Result

You have created a report in your solution and defined the report content using the data defined in your data source. You can check which reports are defined for a particular data source. For more information, see Analytics Quick Guide [page 511] under *Manage Reports*.

You can create a work center and a work center view to which you want your report to be assigned; then, in the UI designer, you can:

1. Add the work center view to the work center.
2. Assign a stable anchor to the work center view to which you want your report to be assigned.
   For more information, see Define an Anchor for a Floorplan [page 753] and Anchors [page 759].
   You must assign this anchor so that administrators of the SAP cloud solution can assign reports to this work center view.
3. Add the *List* and *Gallery* standard reports list views to your work center. These views list all reports that are assigned to the views of the work center.
   For more information, see Add a Report List View to a Work Center [page 752].

> **i Note**
>
> Only administrators for the SAP cloud solution working in a customer's production system can assign a report that you have created to a work center view. To assign a report to a work center view, the solution that contains the report must therefore be implemented in the customer's production system.
>
> We recommend that you provide information about the work center views to which customers need to assign reports in your Solution Documentation. For more information, see Documentation Types in Cloud Solutions from SAP [page 822].

8.11.3.5 Create a Key Figure

In the SAP cloud solution, you can use the administrator tools for Analytics to create a key figure based on an SAP data source or based on a data source you have created (custom data source). This key figure can then be

used in a report. There are two types of key figures: A **calculated key figure** is determined using calculation rules or formulas and a **restricted key figure** is restricted to a specified characteristic value.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio ("studio") as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Prerequisites

- You are assigned the Developer user role and have access rights for the *Business Analytics* work center. For more information, see User Setup Quick Guide [page 33].
- You have switched on the admin mode so that the system saves the analytics content you create in your solution. To do this, select *Administration* > *Switch On/Off Admin Mode in SAP Solution*.
- You have created a data source, if you want to base your key figure on a custom data source. For more information, see Create a Data Source [page 527].

Procedure

1. In the studio, open your solution and, in the *Solution Explorer*, click *Add New Item*.
   The *Add New Item* dialog opens.
2. Select *SAP* > *Analytics* > *Key Figure*.
3. Enter a name and click *Add*.
   The logon screen of the SAP cloud solution opens. Log on to the system.
4. In the *New Key Figure* view, in the *Define Key Figure and Select Data Source* step, if you want, change the default name of your key figure (`AnalyticalKeyFigure1`) and enter a description.
5. Choose an SAP data source or a data source you have created for your solution.
6. Choose which type of key figure you want to create: *Calculated Key Figure* or *Restricted Key Figure*.
7. When you are done, close the SAP cloud solution and go back to the studio.
8. In the studio, in the *Solution Explorer*, update your solution by clicking *Refresh Solution* ( ).

   A `.kf` file ( ) appears under your solution.

Result

You have created a key figure that you can now use in a report. For more information, see Create a Report [page 534].

You can delete the key figure when your solution is in maintenance mode. For more information, see Maintenance Mode [page 185].

8.12 Mass Data Run

8.12.1 Mass Data Run Quick Guide

In the studio, you can create a mass data run (MDR) for your custom business object, define selection parameters for the mass data run, and create the relevant screens. In the SAP cloud solution, your customer uses the mass data run to schedule an action for many instances of a custom business object.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

A mass data run is the automatic mass processing of a task or a business transaction. Mass data runs enable mass processing of business data and are used in business processes, for example, invoice runs, payment authorization runs, or balance confirmation runs. When a user schedules a mass data run, the system executes it as a background job.

For more information, see Mass Data Run [page 538].

Important tasks

Create a Mass Data Run

In the *Solution Explorer*, you can create a mass data run and generate the relevant screens.

For more information, see Create a Mass Data Run [page 542].

Assign MDR Work Center View to UI

In the user interface designer, you can assign the mass data run work center view to the UI.

For more information, see Assign MDR Work Center View to UI [page 544].

Test the Mass Data Run

In the cloud solution, your customer can create and activate mass data run instances and schedule the mass data run.

For more information, see Test the Mass Data Run [page 544].

Related Information

Example: Changing the Status of a Price Request (Mass Data Run) [page 546]

8.12.2 Business and Technical Background

8.12.3 Mass Data Run

With a mass data run (MDR), you can execute an action for many instances of a custom business object without user interaction. You can schedule the mass data run to be executed immediately, on a specific date and at a specific time, or periodically. The node of the custom business object needs to have an action and a query.

> **i** Note
>
> - You cannot use `QueryByElement` queries. These are generated automatically for every node of a custom business object.
> - You cannot create a mass data run for the action of an SAP business object.

For more information, see the mass data run documentation in the SAP cloud solution.

Features

In the studio, you can create a mass data run for a custom business object, which your customer can then use in the SAP cloud solution.

Process Overview

Process steps for you in the studio:

1. In the *Mass Data Run Wizard*: Define the mass data run
    1. Select the custom business object, the node, and the action
    2. Select the query and the selection parameters
    3. Select the control parameters
2. In the *Solution Explorer*: Generate the mass data run screens
3. In the UI designer: Assign the mass data run work center view to the UI

Process steps for a business user in the SAP cloud solution:

1. Create and activate the mass data run instances
2. Schedule the execution of the mass data run

Mass Data Run Wizard

Selection Criteria

In the *Mass Data Run Wizard*, you can use the selection parameters of the query to restrict the business object instances to be processed by the mass data run. You can either select *User Selection* or *Fixed Value*. If you want the business user to be able to restrict the query results by entering values, select *User Selection*. Otherwise select *Fixed Value*; fixed values cannot be changed by the business user.

The following table shows how you define the fixed values for specific data types:

Data Type	Multiple Values	Equal To	Between	Contains Pattern	Less Than/ Less Than Or Equal To	Greater Than/ Greater Than Or Equal To	Format	Example
Amount	Yes	Yes	Yes	No	Yes	Yes	Maximum of 22 integer and 6 decimals places	777.95
Date	Yes	Yes	Yes	No	Yes	Yes	yyyy-mm-dd	2013–04–19
Code	No	Yes	No	No	No	No	String	
Duration	Yes	Yes	Yes	No	Yes	Yes	PnYnMnDTnHnMnS (see below)	P12Y12M2DT4H12M40S
Identifier	Yes	Yes	Yes	Yes	Yes	Yes	String	
Indicator	No	Yes	No	No	No	No	True/False	

Data Type	Multiple Values	Equal To	Between	Contains Pattern	Less Than/ Less Than Or Equal To	Greater Than/ Greater Than Or Equal To	Format	Example
Measure, Quantity	Yes	Yes	Yes	Yes	Yes	Yes	Positive and negative entries are possible	420.5
Name, Description, Text	Yes	Yes	Yes	Yes	Yes	Yes	String	
Numeric, Percent, Ratio	Yes	Yes	Yes	No	Yes	Yes		12.345
Time	Yes	Yes	Yes	No	Yes	Yes	hh:mm:ss	15:30:00
Value	Yes	Yes	Yes	No	Yes	Yes	Integer values	5

The `Duration` format `PnYnMnDTnHnMnS` means:

- `P`: denotes the duration and precedes every duration value
- `nY`: the number prefix (n) is the duration in years
- `nM`: the number prefix (n) is the duration in months
- `nD`: the number prefix (n) is the duration in days
- `t`: the prefix for the time period in hours, minutes, and seconds. This is not required if no times are represented.
- `nH`: the number prefix (n) is the duration in hours
- `nM`: the number prefix (n) is the duration in minutes
- `nS`: the number prefix (n) is the duration in seconds

Parallel Processing

Parallel processing can shorten the processing time of a mass data run. If you select parallel processing, blocks of 50 business object instances are processed in parallel. The number of processes being executed in parallel depends on the current system activities and cannot be influenced. If parallel processing of the business object instances changes the same instance of another business object, locking problems can occur.

> ⚠ Caution
>
> Only use parallel processing if you are sure that the business object instances are processed independently of each other.

Application Log

For each execution of a mass data run, the system creates an application log. This shows the selection criteria used for the mass data run and the messages that the system generated during the execution. In addition, the application log details the number of instances processed and the number of instances changed. When you define the mass data run, you can specify which messages (error, warning, or success) you want the application log to display. Application logs are stored for the retention time you define (one day, one week, or one month). The next time the mass data run is executed, the system deletes application logs for which the retention time has expired.

For more information, see the application log documentation in the SAP cloud solution.

Scheduling of Mass Data Runs in the SAP Cloud Solution

To execute a mass data run, business users have to create a new instance in the MDR work center view. They can also add further selection criteria. After setting the mass data run instance to *Active*, the new mass data run can be scheduled. Business users can choose from the following scheduling options:

- *Start Immediately*
  Immediate execution of the mass data run. Depending on the current system activities, it may take some time until the process starts.
- *Run After Job*
  Execution once another mass data run is completed. This option can be used if there are dependencies between the mass data runs.
- *Single Run*
  Single execution on a specific date or at a specific time.
- *Recurrence*
  Recurring execution of the mass data run (daily, weekly, or monthly).

For more information, see the "Background Jobs Quick Guide" in the SAP cloud solution.

Changing Mass Data Runs

If you change and activate a mass data run during development, the system deletes all existing instances and all scheduled executions runs of that mass data run. This is not the case if a solution with a changed mass data run is uploaded (for example, to a customer system). However, if you change delivered mass data runs, the runtime behavior of scheduled mass data runs may be affected. For example, if you change one of the selection criteria, the list of instances found by the query changes.

You can delete a mass data run when your solution is in maintenance mode. For more information, see Maintenance Mode [page 185].

Mass Data Run in Solution Templates

If you have created a solution template that contains an object work list (OWL) for a mass data run object, you need to do the following: After importing the solution templates into a customer-specific solution, you need to save and activate the OWL again in the UI designer. Otherwise, the system will dump when you create a mass data run instance.

For more information on solution templates, see Solution Templates Quick Guide [page 171].

8.12.4 Tasks

8.12.5 Create a Mass Data Run

In the studio, you define one or more mass data runs (MDR) for your custom business object. First, in the *Mass Data Run Wizard*, you select the relevant business object, node, and action. Then, you select the query and define the selection parameters for the query. After that, you define the control parameters for the execution of the mass data run. Finally, you generate the screens for the mass data run.

Prerequisites

- You have created a solution.
- The solution contains a custom business object with an action. For more information, see the Business Objects Quick Guide [page 398] under **Create a Business Object** and Action (Business Object) [page 194].
- You have created a query for the business object in the node containing the action. For more information, see Create a Query [page 421].
- You have created a screen scenario. For more information, see Generate Screens for a Business Object [page 429].

Procedure

1. In the studio, in the *Solution Explorer*, right-click the custom business object and select *Create Mass Data Run*.
   The *Mass Data Run Wizard* opens.

   > **i Note**
   >
   > If the *Create Mass Data Run* option is disabled (grayed out) in the context menu of the custom business object, the prerequisites for creating a mass data run are not met. This is the case, for example, when the business object does not contain an action or a query.

2. In the *Create a Mass Data Run* step, enter a name, select a business object, a node, and an action, and click *Next*.

   > **i Note**
   >
   > The name of the mass data run can have up to 18 characters.

3. In the *Define Selection Criteria for Mass Data Run* step, do the following:
   1. Select a query and then the restrictions *None*, *User Selection*, or *Fixed Value* for the query fields.

      > **i Note**
      >
      > You must select *User Selection* or enter fixed values for at least one field.

If you select *Fixed Value*, proceed as follows:

1. Under *Fixed Value*, enter a value or click *...*.
   The *Selection of Fixed Values* dialog opens.

   > **i Note**
   >
   > Depending on the data type of the field, the *...* button may not be available.

2. Click the *Equal to* button on the left and select one of the options.
3. Click *Add* to further restrict the values.

   > **→ Tip**
   >
   > For more information on how to restrict the values, see Mass Data Run [page 538] under **Selection Criteria**.

   You can also exclude values by selecting *But Exclude*.
4. Click *OK*.

2. Click *Next*.

4. In the *Select Control Parameters for Mass Data Run* step, do the following:
   1. Optional: Select *Parallel Processing*, if you want to reduce the processing time. However, you cannot use this option, if there are dependencies among the mass data run instances.
   2. Select the retention time of the application log: *Short (1 Day)*, *Medium (1 Week)*, or *Long (1 Month)*. If the retention time is expired, the application log is deleted during the next execution of the mass data run.
   3. Select the level of detail of the application log. You can choose to have errors, errors and warnings, or all details displayed.
   4. Click *Next*.
5. In the *Review* step, review your entries and click *Finish*.

   In the *Solution Explorer*, a `.run` file ( ) appears.
6. To activate the mass data run, right-click the `.run` file and select *Activate*.
7. Right-click the mass data run and select *Create Screens*.
   Three files (`.OWL.uicomponent`, and `.QA.uicomponent`, `.WCVIEW.uiwocview`) are displayed in the *Solution Explorer* under the `.run` file.

   > **→ Tip**
   >
   > If you delete an element (field) with a label from your business object that is used in the mass data run and query, you must do the following to be able to activate the business object again:
   >
   > 1. Open the query, deselect the field and save the query.
   > 2. Then you can activate the business object.

Result

You have defined one or more mass data runs for your custom business object and created the relevant screens. In the UI designer, you can now assign the MDR work center view to the UI.

For more information, see Assign MDR Work Center View to UI [page 544].

8.12.6 Assign MDR Work Center View to UI

Overview

You have created a mass data run and its screens. Now you want to assign the mass data run work center view to the work center of your custom business object.

Procedure

1. In the *Solution Explorer*, open the *_WCF.WCF.uiwoc* file of the custom business object. Right-click the file and select *Open UI Designer*.
   The UI designer opens.
2. On the *Designer* tab, select the *TestView* work center view.
3. In the *Configuration Explorer*, select *[Your Solution]_MAIN* > *SRC* > *[your mass data run work center view]_WCView* and drag and drop it onto the empty space in the middle of the screen.
   A warning message appears.
4. Click *Yes* and then *Save and Activate* and close the UI designer.
5. Reopen the UI designer by right-clicking the *_WCF.WCF.uiwoc* file of the custom business object.
   On the *Designer* tab, the *TestView* work center view should now have changed its name to the name of your mass data run.

> → Tip
>
> If you want to change the texts on the screens, open the relevant screen in the UI designer and enter a new text in the *Property Explorer*. For more information, see Change the Title of a Component [page 739].

Result

You have assigned the mass data run screens to the work center of the custom business object. Your customer can now define and schedule mass data runs in the cloud solution.

For more information, see Test the Mass Data Run [page 544].

8.12.7 Test the Mass Data Run

Overview

You have assigned the mass data run work center view to the UI. Now you can test the mass data run in the SAP cloud solution.

Prerequisites

In the UI designer, you have assigned the mass data run screens to the work center of the custom business object. For more information, see Assign MDR Work Center View to UI [page 544].

In the SAP cloud solution, you have assigned the work center of your custom business object to your business user. For information on how to assign a work center view to a business user, see the User Setup Quick Guide [page 33].

Procedure

1. In the SAP cloud solution, choose the work center view of the custom business object and create some instances of the business object.
2. Choose the work center view of the mass data run and click *New*.
3. Enter a *Run ID* and a *Run Description*.
4. To define selection criteria, proceed as follows:
    1. Click one of the *Selection Criteria* tabs and click *Add Row*.
    2. Click *Inclusion/Exclusion* to include or exclude values.
    3. Select a search pattern and enter a range.
    4. Save your entries and click *Set to Active*.
5. Click *Schedule*.
   The *Schedule Job* view opens.
6. The following scheduling options are available:
    - To start the mass data run immediately, select *Start Immediately* and enter a data and time.
    - To start the mass data run after another job, select a job in the *Scheduled Job Selection* dialog or enter a data and time.
    - To schedule the mass data run as a single run, select *Single Run* and enter a date and time.
    - To schedule a recurring mass data run, select *Recurrence*, enter date and time and the type of recurrence.
7. Click *Save and Close*.

Result

You have scheduled the mass data run, which will be processed according to the selections you have made. In the *Mass Data Run* OWL, under *Execution Details*, the status of your mass data run is displayed.

Once the mass data run has been completed, the process status changes. You can then access the application log by clicking *Application Log ID* for more detailed information.

8.12.8 Example

8.12.9 Example: Changing the Status of a Price Request (Mass Data Run)

This example illustrates a use case for mass data runs for custom business objects. Assume the custom business object **PriceRequest** has a validity period. Once the validity of the price request has expired, we want to change its status to obsolete. This status change can be performed using a periodically scheduled mass data run.

Prerequisites

- You have installed the studio on your PC or laptop. For more information, see Installing SAP Cloud Applications Studio [page 30].
- You have a studio developer user and a business user for the cloud solution. For more information, see User Setup Quick Guide [page 33].
- You have created a solution.

Tasks

Create a Code List Data Type

1. Log on to the studio and, in the *Solution Explorer*, select your solution and click *Add Item*.
2. In the *Add New Item* dialog, select *Code List Data Type*.
3. Enter a name (**`PriceRequestStatus`**) and click *Add*.
   The *Code List Data Type Wizard* opens.
4. In the *Create Code List Data* step, proceed as follows:
   Enter a name (**`PriceRequestStatus`**; the suffix `Code` is added automatically) and a description (**`Price Request Status`**) and click *Next*.
5. In the *Define the Code List Values* step, proceed as follows:
   Enter the following values and then click *Next*:

Value	Description	Constant
NEW	New	CO_NEW
ACTIVE	Active	CO_ ACTIVE
OBSOLETE	Obsolete	CO_OBSOLETE

6. In the *Review* step, review your entries and click *Finish*.
   In the *Solution Explorer*, a file is added to your solution. It is called `PriceRequestStatus.codelist`.
7. Right-click the `PriceRequestStatus.codelist` file and select *Activate*.

Create a Custom Business Object With an Action

1. In the *Solution Explorer*, right-click the project in your solution and select *Add New Item*.
2. In the *Add New Item* dialog box, select the *Business Object* template, enter a name for your business object (**`PriceRequest.bo`**), and click *Add*.
   A new item (`PriceRequest.bo`) appears in the *Solution Explorer* and the business object editor opens.
3. In the editor, enter the following code:

```
import AP.Common.GDT as apCommonGDT;

businessobject PriceRequest raises PriceRequestSetToObsolete {

[Label("Price Request ID")] [AlternativeKey]element PriceRequestID : ID;
[Label("Product ID")]element ProductID : ProductID;
[Label("Description")]element Description : MEDIUM_Description;
[Label("Status")]element Status : PriceRequestStatusCode;
[Label("Start Date")] element ValidtyStartDate : Date;
[Label("End Date")] element ValidtyEndDate : Date;
[Label("Price")]element Price : Amount;
[Label("Discount")]element Discount : Percent;

message PriceRequestSetToObsolete text  "Price Request &1 set to obsolete"  :
ID;

action SetToObsolete raises PriceRequestSetToObsolete;
```

4. Save and activate the business object.
5. Right-click the business object and select *Create Script Files*.
   The *Create Script Files* dialog opens: The `SetToObsolete` action is already selected.
6. Click *OK*.
   The script file editor opens.
7. Enter the following code:

```
import ABSL;

var  today = Context.GetCurrentSystemDate();

if (this.ValidityEndDate < Context.GetCurrentSystemDate() && this.Status !=
PriceRequestStatusCode.CO_OBSOLETE )
{
this.Status = PriceRequestStatusCode.CO_OBSOLETE;
raise  PriceRequestSetToObsolete.Create("S", this.PriceRequestID);
}
```

8. Save and activate the script file.

Create a Business Object Query

1. In the *Solution Explorer*, right-click your business object and select *Create Query*.
   The *Query Wizard* opens.
2. In the *Create a Query* step, enter a *Query Description* and click *Next*.
3. In the *Select Query Fields* step, select all fields and click *Next*.
4. In the *Define Query Parameters* step, click *Next*.
5. In the *Review* step, review your entries and click *Finish*.

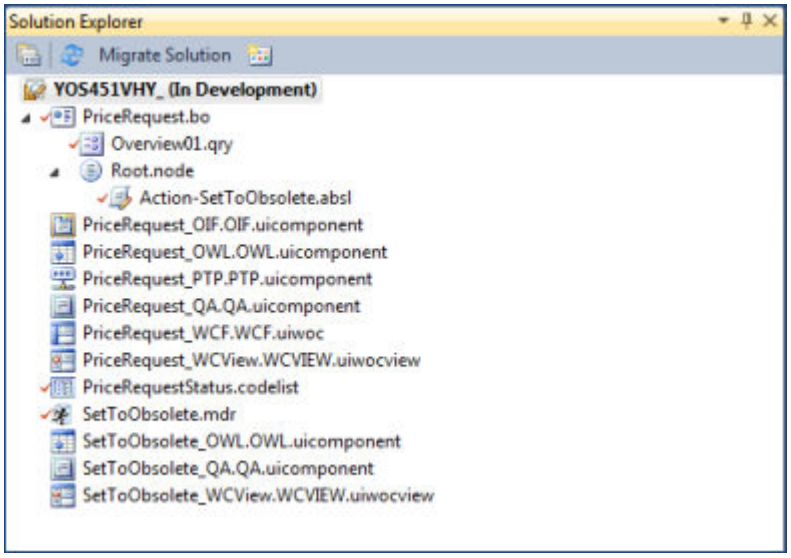
In the *Solution Explorer*, a `.qry` file appears under your business object.

6. Activate the business object.

Create a Screen Scenario for the Business Object

1. In the *Solution Explorer*, right-click the business object and select *Create Screens*.
2. In the *Create Screens* dialog, select a screen scenario option, enter a *Short ID* and click *OK*.
   In the *Solution Explorer*, six new files appear.

> **i Note**
>
> Make a note of the name of your solution (for example, `YOS451VHY`).



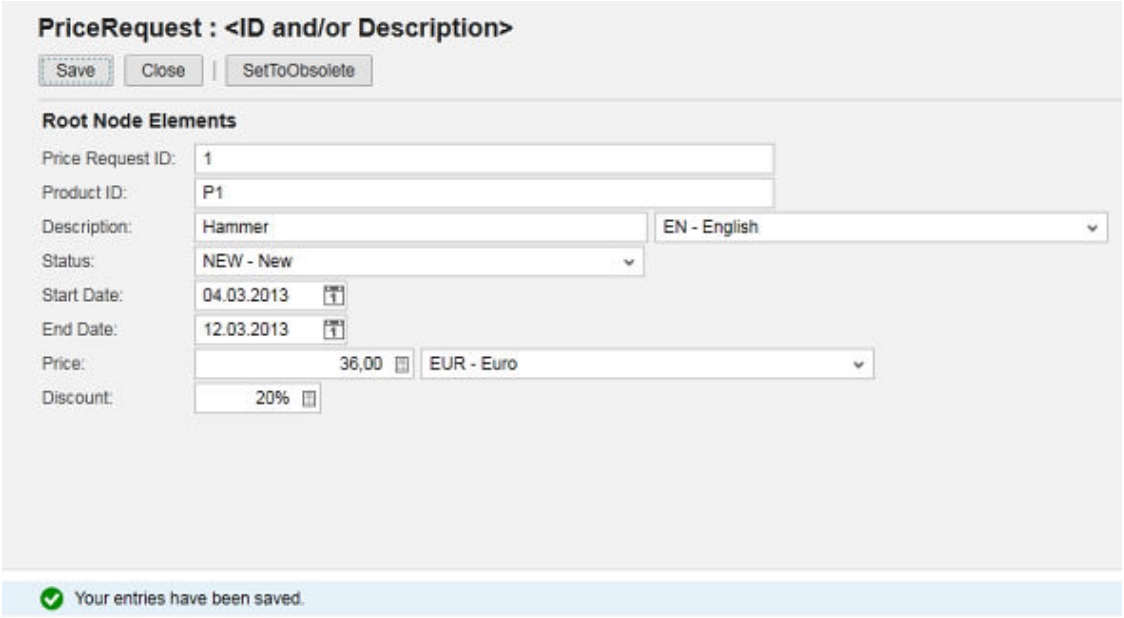
Solution Explorer

Assign the Work Center to a Business User

1. Log on to the cloud solution with an admin user and choose *Application and User Management* > *Business Users*.
2. In the *Find* field, enter the business user and click *Go*.
   The business user is displayed.
3. Select the business user and choose *Edit* > *Access Rights* > *Work Center and View Assignments*.
4. Select the work center you have created (`[your solution name]...WCF.WCF`) and click *Save*.
5. Log off from the cloud solution.

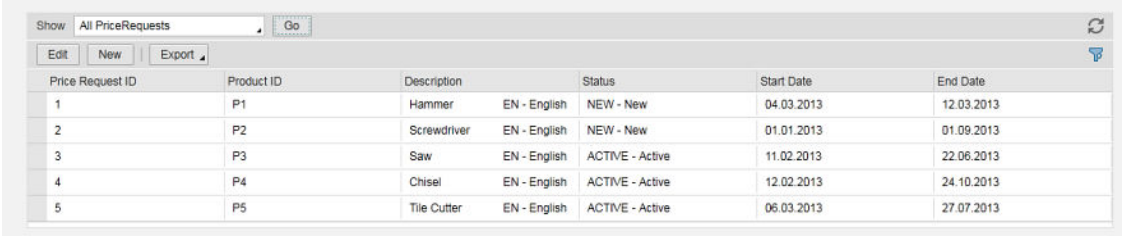
Test the Business Object Action

1. Log on to the cloud solution with the business user and choose the *PriceRequest* work center you have created and assigned to the business user.
2. Create several price requests with different validities:
    1. Click *New*, enter values in all fields and click *Save*.



New Price Request

    2. Click *Close* and repeat step 1.
3. Test the *SetToObsolete* action:
    1. Click *Go* to display all price requests.



All Price Requests

    2. Select one of the entries and click *Edit*.
    3. Change the validity end date to a date before the current date, click *Save* and then *SetToObsolete*. The status of the price request is changed to *Obsolete*.

Create a Mass Data Run

1. In the studio, in the *Solution Explorer*, right-click your business object and select *Create Mass Data Run*. The *Mass Data Run Wizard* opens.
2. In the *Create a Mass Data Run* step, enter a name (for example, **`SetToObsolete`**), check the other values and click *Next*.

3. In the *Define Selection Criteria for Mass Data Run* step, for the `Status: PriceRequestStatusCode` query field, set the restriction to *User Selection* and click *Next*.
4. In the *Select Control Parameters for Mass Data Run* step, leave the settings as they are (*Parallel Processing:* not selected, *Application Log Retention Time:* **`Short (1 Day)`**, *Application Log Detail Level:* **`Errors`**) and click *Next*.
5. In the *Review* step, review your entries and click *Finish*.

   In the *Solution Explorer*, a `.run` file ( ) appears.
6. To activate the mass data run, right-click the `.run` file and select *Activate*.

Create the Screens and Assign Them to the Work Center

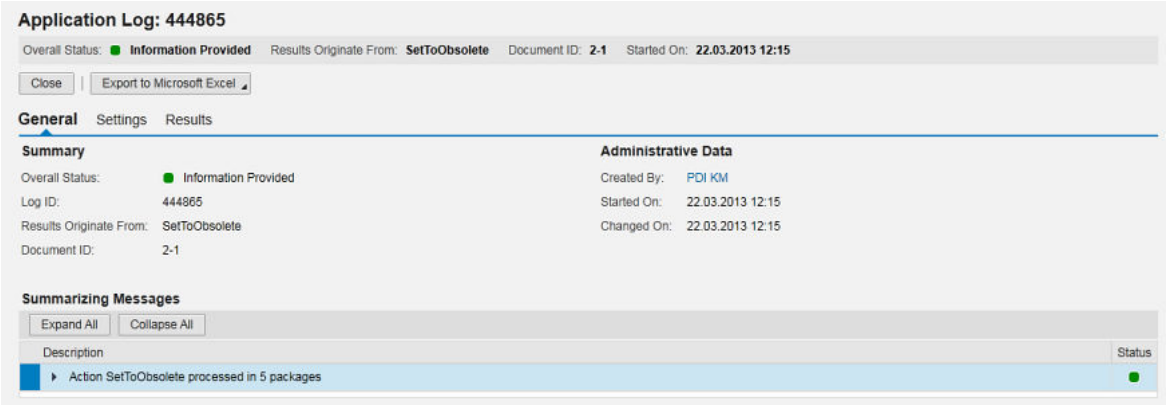
1. In the *Solution Explorer*, right-click the `.mdr` file and select *Create Screens*.
   The system generates three screen files (QA, OWL, WCVIEW).
2. To open the `PriceRequest_WCF.WCF.uiwoc` file, right-click it and select *Open UI Designer*.
   The UI designer opens.
3. On the *Designer* tab, select the *TestView*.
4. In the *Configuration Explorer*, select *[Your Solution]_MAIN* > *SRC* > *SetToObsolete_WCView* and drag it onto the empty space in the middle of the screen.
   A warning appears.
5. Click *Yes* and then *Save and Activate* and close the UI designer.
6. In the cloud solution, assign the `SetToObsolete` work center view to your business user.

Create and Schedule Mass Data Run Instances

1. In the cloud solution, go to the object work list (OWL) of your mass data run and click *New*.
2. Enter a run ID and a run description.
3. To add selection criterion, click *Add Row*.
4. Enter the following data:

Inclusion/Exclusion	Search Pattern	From
Exclusion	EQ — Equal to	OBSOLETE — Obsolete

5. Click *Save* and *Set to Active*.
6. Click *Schedule*.
   The *Schedule Job* view opens.
7. Select *Start Immediately* and click *Save and Close*.
   You have started a mass data run. As soon as the mass data run has finished, you can see the result in the *SetToObsolete* work center view.
   You can call up further details by clicking the *Application Log ID*.



Mass Data Run: Application Log

Related Information

Mass Data Run [page 538]
Mass Data Run Quick Guide [page 537]

8.13 Mashups

8.13.1 Mashups Quick Guide

Using the studio, you can create mashups for an SAP cloud solution, and add them to an SAP floorplan or a floorplan that you have designed.

Business and Technical Background

Creating Mashups

Using the studio, you can create mashups for an SAP cloud solution, and add them to an SAP floorplan or a floorplan that you have designed.

For more information, see here [page 553].

Mashup Categories

For more information, see .

Tasks

Create a Port Binding for a Mashup

For information about this task, see here [page 554].

Create a URL Mashup

For information about this task, see here [page 555].

Create an HTML Mashup

For information about this task, see here [page 558].

Create a Data Mashup

For information about this task, see here [page 556].

Add a Mashup to an SAP Floorplan

For information about this task, see here [page 560].

Add a Mashup to Your Floorplan

For information about this task, see here [page 568].

Save and Activate your Changes

After you have made any of the changes listed above, you need to save and activate the changes.

1. From the *File* menu, choose *Save*.
2. From the *File* menu, choose *Activate*.
3. In the *Activate Worklist* dialog box, select the check boxes beside the relevant components and click *OK*.

8.13.2 Business and Technical Background

8.13.2.1 Creating Mashups

Overview

Using the studio, you can create mashups for an SAP cloud solution, and add them to an SAP floorplan or a floorplan that you have designed.

Prerequisites

The following prerequisites apply:

- You must enable the admin mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, open your solution and select *Administration* > *Switch On/Off Admin Mode in SAP Solution*. For more information about the admin mode, see Working in the Integrated Development Environment [page 42].

If you create a Web service for a mashup, the following prerequisites apply concerning API keys and authentication:

Features

The following mashup types are available:

You can create new data, URL, and HTML mashups. Custom mashups can be created only by SAP.

For more information, see .

Process Flow

The following process steps apply:

1. **Create a Mashup**
   You can create the following mashup types:
   - Create a URL Mashup [page 555]
   - Create an HTML Mashup [page 558]
   - Create a Data Mashup [page 556]
2. **Add the Mashup to a Floorplan**
   After that, you can add the mashup to an SAP floorplan or a floorplan that you have designed. For more information, see:
   - Add a Mashup to an SAP Floorplan [page 560]

- Add a Mashup to Your Floorplan [page 568]

> **i Note**
>
> If you want to integrate a mashup into a floorplan provided by SAP, this floorplan must have an existing outport with a stable anchor and a referenced anchor. You cannot create your own outports for SAP floorplans.

Related Information

Mashups Quick Guide [page 551]

8.13.3 Tasks

8.13.3.1 Create a Port Binding for a Mashup

Overview

The SAP cloud solution provides port bindings that specify on which screens mashups can be used. However, you can also create your own port bindings for your solutions, if required.

Prerequisites

You must have created a solution. The following prerequisites for creating solutions that use mashups apply:

- You must enable the admin mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, open your solution and select *Administration* > *Switch On/Off Admin Mode in SAP Solution*. For more information about the admin mode, see Working in the Integrated Development Environment [page 42].

Before you can create a port binding, you need to create a port type package for the mashup inport. If the mashup is going to return data that is entered automatically into the cloud solution, then you need to create a port type package for the mashup outport as well. For information on creating port type packages, see Create a Port Type Package [page 713].

Procedure

1. Open your solution.
2. In the *Solution Explorer*, under *Mashup Configuration*, right-click *Port Binding* and select *Create Port Binding*.

3. Select the *Mashup Port Binding* template.
4. Enter a name for the port binding and click *Add*.
5. Right-click the port binding, and choose *Open*.
   The UI designer opens, displaying a dialog box that contains the name of the port binding in the title.
6. In the *Display Name* field, enter the name of the port binding. In the *Description* field, enter a text that describes what the port binding is used for.
   This information is displayed in the dropdown list when you create a mashup and select a port binding.
7. In the *Category* field, select the mashup category to which the port binding should be assigned.
8. Select an inport type package from the dropdown list.

   > **i Note**
   >
   > Only your own inport type packages are displayed in the dropdown list.

9. Select an inport type reference.

   > **i Note**
   >
   > Only inport type references that have been defined for the port type package that you have selected are displayed.

10. If the mashup is going to return data to the cloud solution, select also an outport type package and an outport type reference.
11. Save and activate.

Result

You can now use the port binding when you create a mashup. For more information on creating mashups, see Creating Mashups [page 553].

Related Information

Mashups Quick Guide [page 551]

8.13.3.2 Create a URL Mashup

In the studio, you can create URL mashups to send data from the SAP cloud solution to a URL of a Web service provider. The results are displayed on the Web service provider's Web site, which is opened in a new browser window.

Prerequisites

- You must enable the admin mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, open your solution and select *Administration* > *Switch On/Off Admin Mode in SAP Solution*. For more information about the admin mode, see Working in the Integrated Development Environment [page 42].

Procedure

Open your solution. In the *Solution Explorer*, do one of the following steps:

- If you are creating a mashup for the first time for this solution, right-click *Add-on Solution (In Development)*, then select *Add New Item* > *Create URL Mashup*.
- If you have already created a mashup for this solution, you can simply right-click *MashupAuthoring* and select *Create URL Mashup*.

The SAP cloud solution opens. If required, enter your user and password, then do the following steps:

Result

When you return to the studio, click *Refresh* to display the mashup in the *Solution Explorer*.

You can now do the following tasks:

- Add a Mashup to an SAP Floorplan [page 560]
- Add a Mashup to Your Floorplan [page 568]

The document Add a URL Mashup [page 116] provides an example of how you can add a URL mashup to a quick activity floorplan.

Related Information

Mashups Quick Guide [page 551]

8.13.3.3 Create a Data Mashup

In the studio, you can create data mashups for integrating data provided by third-party Web services with business data from the SAP cloud solution.

> **i Note**
>
> Note that some Web services may pass your business data to a third-party organization, for example, account data is passed to a search engine when performing a reverse lookup in an online address book. It is

recommended that you check whether the mashup conforms to your company's data privacy policies before activating the mashup.

Prerequisites

For more information about adding a new RSS/Atom, REST, or SOAP Web service, see Create a Mashup Web Service [page 559].

- You must enable the admin mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, open your solution and select *Administration* > *Switch On/Off Admin Mode in SAP Solution*. For more information about the admin mode, see Working in the Integrated Development Environment [page 42].

Procedure

Open your solution. In the *Solution Explorer*, do one of the following steps:

- If you are creating a mashup for the first time for this solution, right-click *Add-on Solution (In Development)*, then select *Add New Item* > *Create Data Mashup*.
- If you have already created a mashup for this solution, you can simply right-click *MashupAuthoring* and select *Create Data Mashup*.

The SAP cloud solution opens. If required, enter your user and password, then do the following steps:

Result

When you return to the studio, click *Refresh* to display the mashup in the *Solution Explorer*.

You can now do the following tasks:

- Add a Mashup to an SAP Floorplan [page 560]
- Add a Mashup to Your Floorplan [page 568]

Related Information

Mashups Quick Guide [page 551]

8.13.3.4 Create an HTML Mashup

Overview

In the studio, you can create HTML mashups to embed an HTML or JavaScript based Web page into a screen.

Prerequisites

- You must enable the admin mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, open your solution and select *Administration* > *Switch On/Off Admin Mode in SAP Solution*. For more information about the admin mode, see Working in the Integrated Development Environment [page 42].

Procedure

Open your solution. In the *Solution Explorer*, do one of the following steps:

- If you are creating a mashup for the first time for this solution, right-click *Add-on Solution (In Development)*, then select *Add New Item* > *Create HTML Mashup*.
- If you have already created a mashup for this solution, you can simply right-click *MashupAuthoring* and select *Create HTML Mashup*.

The SAP cloud solution opens. If required, enter your user and password, then do the following steps:

Result

When you return to the studio, click *Refresh* to display the mashup in the *Solution Explorer*.

You can now do the following tasks:

- Add a Mashup to an SAP Floorplan [page 560]
- Add a Mashup to Your Floorplan [page 568]

Related Information

Mashups Quick Guide [page 551]

8.13.3.5 Create a Mashup Web Service

Overview

In the studio, you can create mashup Web services that can then be integrated into a data or HTML mashup. RSS/Atom, REST, SOAP, and OData Web services are supported.

Prerequisites

- For RSS or Atom feeds, you know the end-point URL of the Web service or the WSDL location.
- For REST Web services, you need to know the request URL as well as any relevant parameters.
- For SOAP Web services, you need to know the WSDL location (URL).

Typically, you can find this information on the Web site of the service provider.

> **i** Note
>
> You can also integrate an external SOAP or REST Web service into your solution. To do this, you use the *External Web Service Integration Wizard*.
>
> For more information, see Integrate an External Web Service Using SOAP [page 612] and Integrate an External Web Service Using REST [page 614] .

Procedure

Open your solution. In the *Solution Explorer* do one of the following steps:

- If you are creating a mashup Web service for the first time for this solution, right-click *Add-on Solution (In Development)*, then select *Add New Item* > *Create .... Mashup Web Service* (for example, *Create REST Mashup Web Service*).
- If you have already created a mashup Web service for this solution, you can simply right-click *MashupAuthoring* and select *Create .... Mashup Web Service* (for example, *Create REST Mashup Web Service*).

The SAP cloud solution opens. If required, enter your user and password, then do the following steps:

Result

Related Information

Using a Mashup Web Service in a Script File [page 563]
Mashups Quick Guide [page 551]

8.13.3.6 Add a Mashup to an SAP Floorplan

Overview

When you add a mashup to an SAP floorplan, you can specify how it should appear on the screen. The following options are available:

- As an embedded pane
  This pane displays the mashup content directly in a section on the screen.
- As a new tab
- As a menu button
  The mashup can be accessed from *Web Services* located at the top of a screen.
- As a link in a section of a screen
  The link opens the mashup in a separate browser window.

The options that are available depend on the mashup itself and the anchors that the floorplan has. The following table lists the anchors and the possible appearances on the screen.

Anchors and Appearance on Screen

Add Mashup Through	Appearance on Screen
Floorplan anchor	Embedded pane on a section of a screen Menu button
Outport anchor	Embedded pane on a section of a screen Tab Menu button Link in a section of a screen
Section group anchor	Embedded pane on a section of a screen Link in a section of a screen
Switch navigation anchor	Tab
Overview page anchor	Embedded pane on a section of a screen

You can also choose an extension field to allow the user to toggle the display of the mashup on the screen by selecting and deselecting this extension field.

Prerequisites

- You have created the mashup. For more information, see:
  - Create a URL Mashup [page 555]
  - Create an HTML Mashup [page 558]

- Create a Data Mashup [page 556]

- You must enable the admin mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, open your solution and select *Administration* > *Switch On/Off Admin Mode in SAP Solution*. For more information about the admin mode, see Working in the Integrated Development Environment [page 42].
- The SAP floorplan must have an existing outport with a stable anchor and a referenced anchor. You cannot create your own outports for SAP floorplans.

Procedure

1. **Open the SAP floorplan.**
   1. In the studio, right-click one of your floorplans in your solution, and select *Open in UI Designer*.
      The UI designer opens.
   2. In the UI designer, go to *Configuration Explorer*, and navigate to the SAP floorplan in which you want to add the mashup.
   3. Right-click the SAP floorplan, and click *Open*.
      The SAP floorplan opens in the *Designer* tab.
2. **Add the mashup to the SAP floorplan.**
   You can do this in the following ways:
   Add Mashup Through a Floorplan Anchor
   1. In the *Configuration Explorer*, open an SAP floorplan that has a floorplan anchor.
   2. Right-click the floorplan in the *Designer* tab, and choose *Select* > *Floorplan*.
   3. In the *Extensibility Explorer*, under *Operations*, click *Widget Mashups Management*.
      A dialog box opens, and a list of available mashups is displayed.
   4. Select the required mashup.
      The configuration details are displayed below.
   5. Select the checkbox in the *Visible* column. This allows the mashup to be displayed on the screen.
   6. Select the required appearance in the *Appearance* field.
   7. If a mashup should be displayed using the entire width of the screen, select *Full Screen Width*.
   8. If there are extension fields of the type boolean, you can select an extension field in the **Visibility Binding** field, to allow the user to toggle the visibility of the mashup on the screen
   9. Click *Apply*.
      The system creates a change transaction.
   10. Save and activate your changes.

   Add a Mashup Through an Outport Anchor
   1. In the *Configuration Explorer*, open an SAP floorplan that has an outport anchor.
   2. In the *Controller* tab, choose an outport under the *Outports* node.
   3. In the *Extensibility Explorer*, under *Operations*, click *Mashups Management*.
      A dialog box opens, and a list of available mashups is displayed.
   4. In the *Visible* column, select the checkbox for each mashup that you want to add to the floorplan.

      > **i Note**
      >
      > If the *Visible* column for a mashup is disabled, it means that the mashup is already available on the floorplan. To remove the mashup, you need to delete the corresponding change transaction in *Extensibility Explorer*.

5. Select a mashup to configure it.
   The configuration details are displayed below.
6. Select the required appearance in the *Appearance* field.
7. If the mashup should be displayed using the entire width of the screen, select *Full Screen Width*.
8. If there are extension fields of the type boolean, you can select an extension field in the **Visibility Binding** field, to allow the user to toggle the visibility of the mashup on the screen
9. Click *Apply*.
   The system creates a change transaction.
10. Save and activate your changes.

Add a Mashup Using a Section Group Anchor

1. In the *Configuration Explorer*, open an SAP floorplan that has a section group anchor.
2. Select a section group in the floorplan.
3. In the *Extensibility Explorer*, under *Operations*, click *Mashups Management*.
   A dialog box opens, and a list of available mashups is displayed.
4. In the *Visible* column, select the checkbox for each mashup that you want to add to the floorplan.

> **i** Note
>
> If the *Visible* column for a mashup is disabled, it means that the mashup is already available on the floorplan. To remove the mashup, you need to delete the corresponding change transaction in *Extensibility Explorer*.

5. Select a mashup to configure it.
   The configuration details are displayed below.
6. Select the required appearance in the *Appearance* field.
7. Click *Apply*.
   The system creates a change transaction.
8. Save and activate your changes.

Add a Mashup Through a Switch Navigation Anchor

1. In the *Configuration Explorer*, open an SAP floorplan that has a switch navigation anchor.
2. Right-click the floorplan in the *Designer* tab, and choose *Select* > *NavigationItem*.
3. In the *Extensibility Explorer*, under *Operations*, click *Mashups Management*.
   A dialog box opens, and a list of available mashups is displayed.
4. Select the required mashup.
   The configuration details are displayed below.
5. Select the checkbox in the *Visible* column. This allows the mashup to be displayed on the screen.

> **i** Note
>
> If the *Visible* column for a mashup is disabled, it means that the mashup is already available on the floorplan. To remove the mashup, you need to delete the corresponding change transaction in *Extensibility Explorer*.

6. Select the required appearance in the *Appearance* field.
7. If a mashup should be displayed using the entire width of the screen, select *Full Screen Width*.
8. If there are extension fields of the type boolean, you can select an extension field in the **Visibility Binding** field, to allow the user to toggle the visibility of the mashup on the screen
9. Under Parameter Binding, select the required extension fields to be bound to the parameters.

> **Note**
>
> Parameters that are marked with an asterisk (*) are mandatory.

10. If the mashup has an output parameter, select an extension field for output binding.
11. Click *Apply*.
    The system creates a change transaction.
12. Save and activate your changes.

Add a Mashup Through an Overview Page Anchor

1. In the *Configuration Explorer*, open an SAP floorplan that has an overview page anchor.
2. In the *Extensibility Explorer*, under *Operations*, click *Add Content to Overview*.
   A dialog box opens.
3. In the *Type* field, select `Mashup` from the dropdown list.
   The dialog box changes.
4. In the *Mashup* field, select the mashup that you want to add to the floorplan from the dropdown list, and click *Apply*.

   > **Note**
   >
   > If the selected mashup is already part of the view, the *Apply* button is disabled.

5. If the mashup should be displayed using the entire width of the screen, select *Full Screen Width*.
6. Save and activate your changes.

Result

The mashup is now visible on the floorplan.

> **Note**
>
> To remove a mashup from an SAP floorplan, you need to delete the corresponding change transaction in the *Extensibility Explorer* of the UI designer, under *Operations*.

Related Information

Mashups Quick Guide [page 551]

8.13.3.7 Using a Mashup Web Service in a Script File

Overview

If you have configured a REST or RSS/Atom mashup Web service, you can implement the call of this Web service in a script file of your solution.

Prerequisites

- You have defined an active REST or RSS/Atom mashup Web service in your solution, either with basic authentication or without authentication.
- You have created a solution with at least one business object. In the example below, an action has been added to the business object definition.

Process Flow

1. Create a REST or RSS/Atom mashup Web service in your solution.
   For more information, see Create a Mashup Web Service [page 559].
   After you have created the mashup Web service, choose *Simulate* in the *Mashup Web Services* view to test it.
2. Create a script file for your business object, for example, an action.
3. Define the business logic to call the mashup Web service. The following is an example of an action.

   Example

   ```
   import AP.Common.GDT;
   ```

   ```
   var serviceID : LANGUAGEINDEPENDENT_Text;
   var parameter  : NameAndValue;
   var parameters : collectionof NameAndValue;
   ```

   ```
   serviceID = "PW00001";
   parameter.Name = "keywords";
   parameter.Value = "Rome";
   parameters.Add(parameter);
   ```

   ```
   var result = WebServiceUtilities.ExecuteWebService(serviceID, parameters);
   var content = result.ResponseContent;
   var returnCode = result.ReturnCode;
   ```

   The parameters of this Web service call are as follows:
   - Input Parameters
     - Service ID (mandatory)
       The ID of the REST or RSS/Atom mashup Web service of type `String`. Do not include the solution namespace. For example, if the service ID is displayed as follows: `YUYYRY2MY_MAIN/PW00001`, use only `PW00001` in the call.
     - Parameters (optional)
       Parameter of type `collectionof NameAndValue`. This refers to the table of input parameters defined in the mashup Web service.
   - Result
     The type is `WSCallResult` and contains the following elements:
     - `ReturnCode`
       The return code of the Web service call as `String`.
     - `ResponseContent`

The content of the response as `String`.

4. Save your script file.

8.13.3.8 Add a URL Mashup to an SAP Floorplan

Overview

In this example you are going to add a URL mashup to the quick activity (QA) floorplan. This mashup will display the car park location on a map, based on the location information entered.

Prerequisites

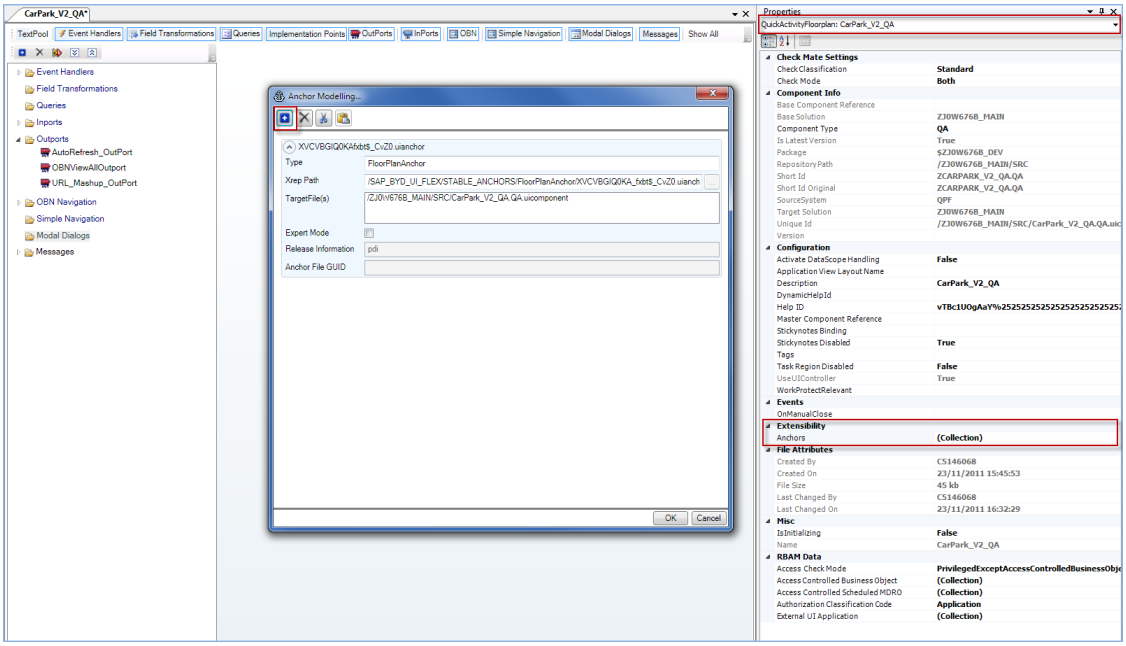
- You have completed the example: Building a Solution [page 99]
- The mashup has been created in the SAP cloud solution. For more information, see Mashups Quick Guide [page 551].

Procedure

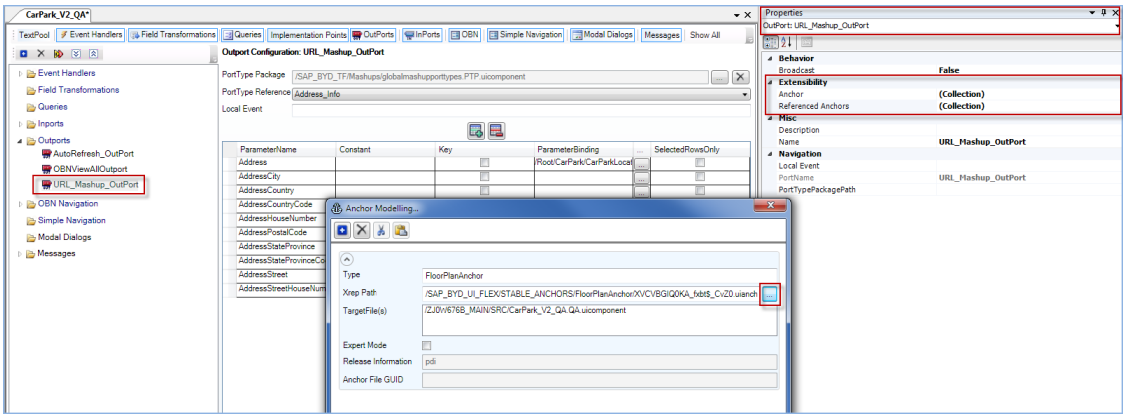
1. Add an element on the root level of the Car Park business object to record location information.

```
element CarParkLocation : LANGUAGEINDEPENDENT_Text;
```

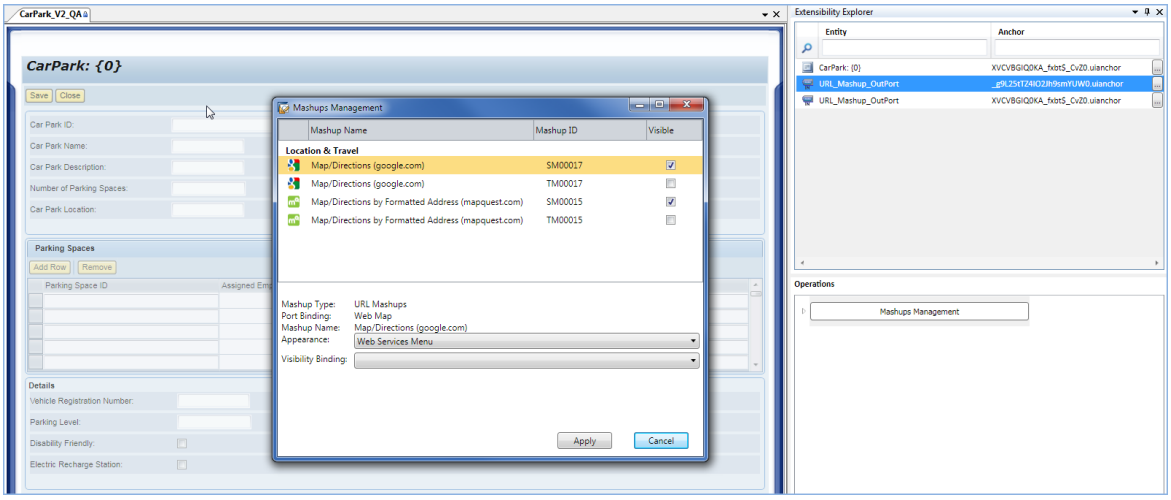
2. Save and activate the business object.
3. Double-click the quick activity floorplan to open it in the UI designer.
4. Click the *Update Metadata* button to update the BO metadata.
5. Drag the *CarParkLocation* from the *BO Browser / Data Model* onto the *Designer* tab.
6. Change the label for the field to **`Car Park Location`**.
7. On the *Controller* tab, perform the following actions:
   1. Add a new outport with the name **`URL_Mashup_OutPort`** and the following settings:
      - *PortType Package* = `/SAP_BYD_TF/Mashups/globalmashupsupporttypes`
      - *PortType Reference* = `Address_Info`
   2. Bind the *Address* parameter to the *CarParkLocation* data field.
   3. In the *Property Explorer*, select the QA floorplan from the drop-down list at the top.
   4. Click the *Anchor* field under *Extensibility*.
      The *Anchor Modelling* dialog box appears.
   5. Click the plus sign to create a stable anchor for the floorplan.



6. In the *Property Explorer*, select the mashup outport from the drop-down list at the top.
7. Click the *Anchor* field under *Extensibility*.
   The *Anchor Modelling* dialog box appears.
8. Click the plus sign to create a stable anchor for the floorplan.
9. In the *Property Explorer*, click the *Referenced Anchor* field under *Extensibility*.
   The *Anchor Modelling* dialog box appears.
10. Click the plus sign to create a referenced anchor for the outport.
11. Click the three dots next to the *XRep Path* field.
12. In the *Configured Anchors* dialog box, click the anchor you created for the QA floorplan.



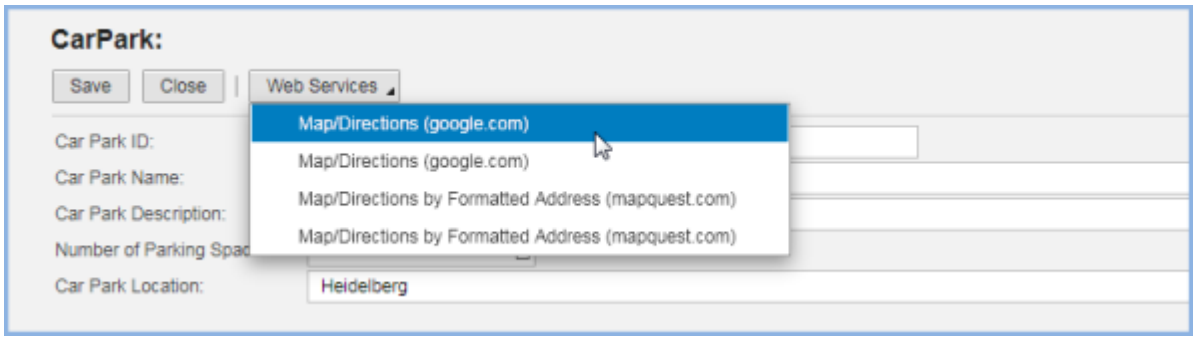
8. Save and activate.
9. Click the *Designer* tab.
10. In the *Extensibility Explorer*, select the mashup outport and click the *Mashups Management* button.
    The *Mashups Management* dialog box appears, listing the available mashups.
11. Select the mashup you want to use and click *Apply*.

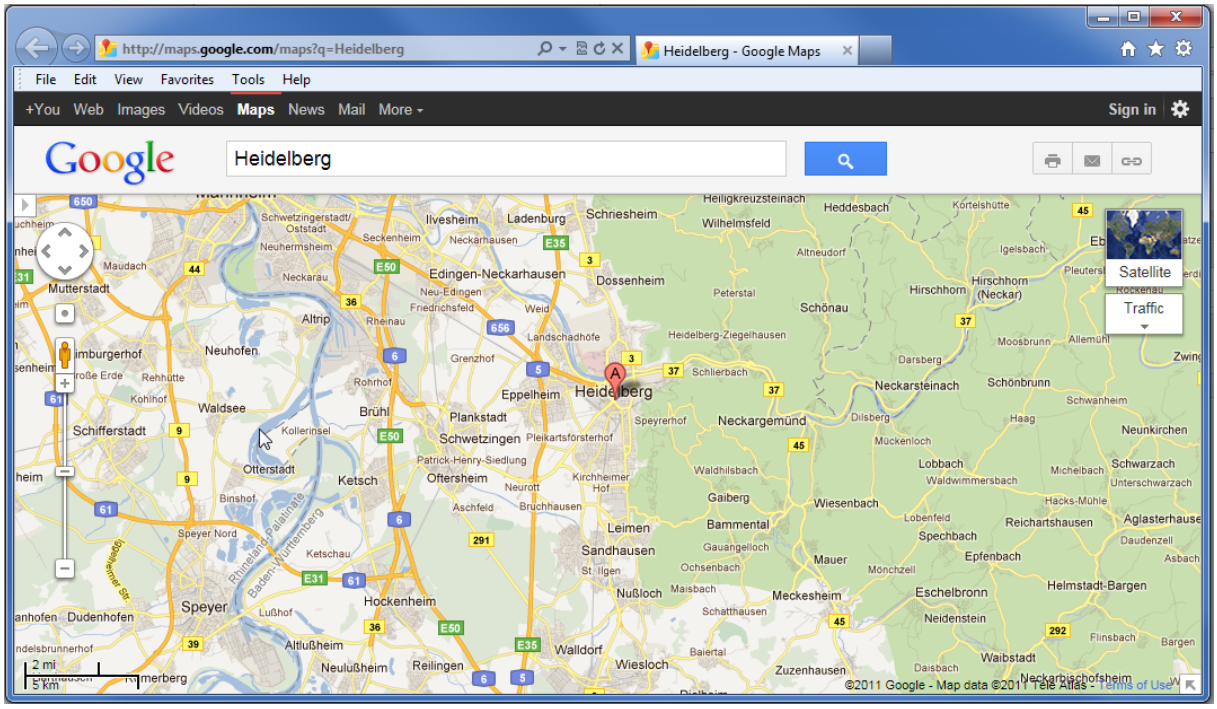


12. Save and activate.

Result

The mashup is available from the *Web Services* menu on the QA. When you click the mashup, the URL is opened with the value entered into the *Car Park Location* field as an input parameter.





8.13.3.9 Add a Mashup to Your Floorplan

Overview

When you add a mashup to a floorplan that you have designed, you can specify how it should appear on the screen. The following options are available:

- As an embedded pane, with or without a port
  This pane displays the mashup content directly in a section on the screen.
- As a new tab
- A menu button in a function bar
  The mashup can be accessed from *Web Services* in the function bar at the top of a floorplan.
- A menu button in a list toolbar
  The mashup can be accessed from *Web Services* in the toolbar at the top of an object worklist.
- A link in a section of a screen
  This link opens the mashup in a separate browser window.

The options that are available depend on the mashup type. The following table lists the mashups and the possible appearances on the screen.

Mashups and Appearance on Screen

Mashup Type	Appearance on Screen
Data mashups that write data back into the SAP cloud solution	Embedded pane on a section of a screen, with or without a port
URL mashups	Menu button to a function bar
	Menu button to a list toolbar
	Link in a section of a screen
Data mashups that do not write data back into the SAP cloud solution	Embedded pane on a section of a screen
HTML mashups	

Prerequisites

- You have created the mashup. For more information, see:
  - Create a URL Mashup [page 555]
  - Create an HTML Mashup [page 558]
  - Create a Data Mashup [page 556]
- You must enable the admin mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, open your solution and select *Administration* > *Switch On/Off Admin Mode in SAP Solution*. For more information about the admin mode, see Working in the Integrated Development Environment [page 42].

Procedure

1. **Open your floorplan.**
   1. In the studio, in the *Solution Explorer*, right-click your floorplan and select *Open in UI Designer*.
      The UI designer opens.
2. **Add the mashup to your floorplan.**
   Choose one of the following options to add a mashup to your floorplan:
   Add a Mashup as a Link / Button in a Function Bar or a List Toolbar / Section of a Screen
   1. In the *Controller*, right-click *Mashup Components*, and select *Add Mashup Component*.
      A dialog box opens.
   2. In the dialog box, go to the main folder of your solution and the folder where the mashups are stored.
   3. Select a URL mashup or a data mashup that writes data back into the SAP cloud solution, and click *OK*.
   4. Go to the *Toolbox* tab to add a navigation item for a function bar or a list toolbar, or as a link to a section group.

      > **i Note**
      >
      > If the *Toolbox* tab is not visible, make it visible by going to *View* > *Toolbox*

5. In the *Controls* section, drag and drop the required element (*Link* / *Button* / *SectionGroup*) to the floorplan.
6. Add an event argument with argument name **`$MashupPipeID`**, and a constant the target mashup component ID, for example,**`TM00020`**.
7. Create a new event handler, and add two operations: *SyncDataContainer* and *FireOutport*.
8. For *FireOutport*, select an available outport such as *Address_Info_Out*. If there is no outport for this floorplan, add a new outport in the *Controller*.
9. On the *Properties* tab, under *Events*, assign the event handler to a navigation item, for example, *OnClick*.
10. In the *Simple Navigation* tab, right-click on the *Navigations* folder, and select *Add Navigation* from the context menu.
11. Select the outport and the inport from the dropdown lists.
12. Bind the outport and inport parameters.

> **i Note**
>
> The outport that you select here should be the same outport that you selected for *FireOutport* a few steps previously.

13. Save and activate your changes.

Add a Mashup With Port as an Embedded Pane

1. In the *Configuration Explorer*, select a mashup that is a an embedded pane, and drag it to a section of the floorplan.
2. In the *Controller*, right-click *Mashup Components*, and select *Add Mashup Component*.
   A dialog box opens.
3. In the dialog box, go to the main folder of your solution and the folder where the mashups are stored.
4. Select a mashup that has the same pipe ID as the one you have just selected, and click *OK*.
5. In the *Simple Navigation* tab, right-click on the *Navigations* folder, and select *Add Navigation* from the context menu.
6. Select the outport and the inport from the dropdown lists.
7. If your floorplan does not have an outport, add an outport as follows:
   1. On the *Controller* tab, right-click *Outports*.and choose *Add Outport*.
      The new outport is displayed under *Outports*.
   2. In the *Port Type Package* field, click ... to select a port type package.
   3. In the dialog box, go to *SAP_BYD_TF* > *Mashups* , choose **`globalmashupporttypes`**, and click *OK*.
   4. In the *Port Type Reference* field, select a port type reference from the dropdown list.
   5. If there are parameters displayed in the table below, you can bind them. To do this, click ... next to *Parameter Binding*.

      > **i Note**
      >
      > If you do not bind the parameters, you will not be able to pass the values from the floorplan to the mashup.

   6. In the dialog box, choose the required element of the business object and click *OK*.
8. Bind the outport and inport parameters.
9. Save and activate your changes.

Add a Mashup Without Port as an Embedded Pane

1. In the *Configuration Explorer*, select a mashup that is an embedded pane, and drag it to a section of the floorplan.
2. Save and activate your changes.

Add a Mashup as a Tab

1. Right-click the floorplan, and choose *Select* > *NavigationItem*.
2. Under *Misc*, click ... next to *NavigationItems* to configure the properties.
3. In the dialog box, select a view and click *Add*.
4. On the new view, add a mashup with or without a port as explained in the previous sections.
5. Save and activate your changes.

Result

You can now test the mashup in your floorplan by clicking the *Run the Component* button (green triangle) in the toolbar.

Related Information

Example: Add a URL Mashup to Your Floorplan [page 571]
Mashups Quick Guide [page 551]

8.13.3.10 Example: Add a URL Mashup to Your Floorplan

Overview

This example shows how to add a URL mashup as a link in a floorplan that you have designed.

Prerequisites

- You have created the mashup. For more information, see:
  - Create a URL Mashup [page 555]
  - Create an HTML Mashup [page 558]
  - Create a Data Mashup [page 556]
- You must enable the admin mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, open your solution and select *Administration* > *Switch On/Off Admin Mode in SAP Solution*. For more information about the admin mode, see Working in the Integrated Development Environment [page 42].

Procedure

1. **Add a link to your floorplan.**
   1. In the studio, in the *Solution Explorer*, right–click your floorplan and select *Open in UI Designer*.
      The UI designer opens.
   2. Go to the *Toolbox* tab.

      > **i** Note
      >
      > If the *Toolbox* tab is not visible, make it visible by going to *View* > *Toolbox*

   3. In the *Controls* section, drag and drop *Link* to the floorplan.
      The *Link* element is displayed on the floorplan. It contains a *Link* field and a *Link* hyperlink.
   4. You can rename the text of the *Link* field. To do this, ensure that the *Link* element is selected.
   5. On the *Properties* tab, under *Text Information*, click on the dropdown arrow in the *Label* field.
   6. In the dialog box, enter the required text, and click *OK*.
      The renaming of the hyperlink will be described later on in this document.
2. **Add your mashup.**
   1. In the UI designer, go to the *Controller* tab of the floorplan.
   2. Right-click on *Mashup Components*, and select *Add Mashup Component* from the dropdown list.
   3. In the dialog box, go to the main folder of your solution and the folder where the mashups are stored.

      > **i** Note
      >
      > This name starts with *MAIN_<Solution Name>* > *SRC* > *Mashups* > *Pipes* > *<Pipe ID>*

   4. Select your mashup as follows.
      - A URL mashup that is displayed in a new window
      - A data mashup that writes data back into the solution
   5. Click *OK*
      The mashup is displayed on the *Controller* tab underneath *Mashup Components*.
3. **Add an event.**
   1. In the UI designer, on the *Designer* tab, select the *Link* element in the floorplan.
   2. On the *Properties* tab, under *Data Information*, select *EventArguments* and click *...*
   3. Click the *Add Argument* button.
      A new event argument is added.
   4. Under *ArgumentName*, rename the event argument to **`$MashupPipeID`**.

      > → Tip
      >
      > We recommend that you copy the name **`$MashupPipeID`** from this document and paste it in the *ArgumentName* field.

   5. Leave the dialog box open, and go to the .
   6. In the , on the *Solution Explorer* tab, under *MashupConfiguration* > *Mashup Authoring*, select the mashup.
   7. On the *Properties* tab, copy the value in the *Repository* field. This value has the format **`MAIN_Solution Name>/<Pipe ID>`**.
   8. Go back to the open dialog box in the UI designer, paste the value of the *Repository* field under *Constant*, and click *OK*.

9. Save your changes.

4. **Create a new outport.**
    1. On the *Controller* tab, right-click *Outports*.and choose *Add Outport*.
    The new outport is displayed under *Outports*.
    2. Right-click the outport and select *Rename*; then enter a new name for the outport.
5. **Select a port type package for the outport.**
    1. In the *Port Type Package* field, click *...*
    2. In the dialog box, go to *SAP_BYD_TF* > *Mashups* >, choose **`globalmashupporttypes`**, and click *OK*.
6. **Configure a port type reference for the outport.**
    1. In the *Port Type Reference* field, select a port type reference from the dropdown list.
    2. If there are parameters displayed in the table below, you can bind them. To do this, click *...* next to *Parameter Binding*.

        > **i** Note
        >
        > If you do not bind the parameters, you will not be able to pass the values from the floorplan to the mashup.

    3. In the dialog box, choose the required element of the business object and click *OK*.
7. **Create an event handler.**
    1. On the *Controller* tab, right-click *Event Handlers* and select *Add EventHandler.*.
    The new event handler is displayed under *Event Handlers*.
    2. Right-click the new event handler and select *Rename*; then enter a new name for the event handler, for example, **`MashupHandler`**.
    3. Under *EventHandler Operations*, click the *Add Operation* button twice to add two new operations.
    4. For each operation, under *Type*, click in the field and select the following operation types:
        - *SyncDataContainer*
        - *FireOutport*
    5. Under *Name*, you can change the names of these operation types
    6. For *FireOutport*, under *Configure the operation*, choose the outport that you created previously.
    7. Save and activate your changes.
8. **Specify the navigation for the mashup.**
    1. On the *Controller* tab, under *Mashup Components*, select the mashup.
    2. Go to the *Simple Navigation* tab.
    3. In the *Navigations* folder, right-click and select *Add Navigation* from the context menu.
    The *Navigation Details* tab opens.
    4. Select the outport and the inport from the dropdown lists.

        > **i** Note
        >
        > The source is the outport of your floorplan; the target is the inport of the mashup category.

    5. To map the outport and inport parameters from the floorplan to the mashup, select these, and click *Bind*.
    The results are displayed in the table below.
    6. Save and activate your changes.
9. **Assign the event handler to the mashup in your floorplan**

1. Go to the *Designer* tab.
2. Select the *Link* element in the floorplan.
3. On the *Properties* tab, under *Events*, select *OnClick*, and select the event handler that you created previously from the dropdown list.

10. **Add a data field to your business object.**
    1. Go to the *Data Model* tab.
    2. Right-click on your business object, and select *Add Data Field* from the context menu.
       The data field is added to your business object.
    3. Select *DataField*.
    4. On the *Properties* tab, under *Design*, enter a text for *Initial Value*.
       This text is used for the hyperlink that is displayed on the floorplan.

       > **i** Note
       >
       > This text is not displayed in the floorplan in the UI designer, but you will see it when you test the mashup.

11. **Assign the data field to the mashup on your floorplan**
    1. Go to the *Designer* tab.
    2. Select the *Link* element.
    3. In the *Properties* tab, under *Data Information*, click the dropdown arrow next to *Text*.
    4. In the dialog box, go to the *Data Model* tab.
    5. Select *DataField* underneath your business object, and click *OK*.
12. **Save and activate your changes.**

Result

You can now test the mashup in your floorplan by clicking the *Run the Component* button (green triangle) in the toolbar.

Related Information

Add a Mashup to Your Floorplan [page 568]
Mashups Quick Guide [page 551]

8.14 Service Integration

8.14.1 Service Integration Quick Guide

In the studio, you can create synchronous or asynchronous message-based communication between a business object and another communication partner, such as a different system, by means of Web-service technology.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Business and Technical Background

Web Services Quick Guide

The studio allows you to create a Web service on an SAP business object or on any of your business objects, which can then be consumed by a client application. This enables remote access to the business object data. You can integrate an external Web service into your solution and address it in your business logic to enable access from the SAP cloud solution to a remote application. You can also select a set of inbound and outbound services to create a communication scenario. This allows customers to create communication arrangements in the SAP cloud solution to exchange business documents electronically. Communication for Web services is based on SOAP or REST.

For more information, see Web Services Quick Guide [page 586].

File Input Quick Guide

In the SAP cloud solution, the *File Input Run* view in the *Application and User Management* work center enables administrators to create and schedule mass data runs for third-party data integration uploading files from other systems. The file input process allows business documents containing the files to be imported without the need for application-specific functionality. During a run, all files of the same type are input. The file input run receives inbound messages containing the files via application-to-application communication.

For more information about this tool, see the relevant documentation for the SAP cloud solution.

Condition Evaluation for Internal Communication

Condition evaluation is used to determine when a business object needs to send a message to the target business object, for example, a message to create or change a business object instance. For each business object identified as relevant for the subsequent service integration process, the system can evaluate a start condition, change condition and a cancel condition. By default, the system sends a message every time a business object is changed or deleted. To reduce the amount of messages, you can define that a message is raised only, for example, when the source business object has a specific status.

For more information, see Condition Evaluation for Internal Communication [page 577].

Tasks

Configure Internal Communication

You can use internal communication to exchange data between business objects that are located in different deployment units (DUs). To communicate, the business objects exchange XML messages in an asynchronous way which means that the source business object sends a request message to the target business object without expecting a response.

For more information about this task, see Configure Internal Communication [page 579].

Create XML File Input

In the SAP cloud solution, administrators can use an XML file to import third-party data from other systems into a business object. To enable this functionality, you create an XML schema definition in the studio. This schema definition allows customers to create an XML file, which can then be used to import the data into the SAP cloud solution.

For more information about this task, see Create XML File Input [page 581].

Define Service Integration Error Handling

Error and conflict handling is needed in message-based internal communication because there is no direct interaction with the user who initiated the process. Thus the receiver of a message has to handle the problem and has to trigger an appropriate resolution mechanism. A major part of development is to avoid error and conflict situations by design but it is not possible to avoid these situations entirely. This means that you also need to consider error and conflict situations in asynchronous communication.

For more information about this task, see Define Service Integration Error Handling [page 583].

Implement the Condition Evaluation Logic for Internal Communication

For internal communication, you can implement the business logic for condition evaluation to specify when a business object needs to send a message to the target business object, for example, a message to create or change a business object instance.

For more information about this task, see Implement the Condition Evaluation Logic for Internal Communication [page 585].

8.14.2 Condition Evaluation for Internal Communication

Overview

Condition evaluation is used to determine when a business object needs to send a message to the target business object, for example, a message to create or change a business object instance. For each business object identified as relevant for the subsequent service integration process, the system can evaluate a start condition, change condition and a cancel condition. By default, the system sends a message every time a business object is changed or deleted. To reduce the amount of messages, you can define that a message is raised only, for example, when the source business object has a specific status.

Features

If you implement the evaluation logic for a condition, the system returns a Boolean value to express if the condition is true or false, for example, based on the status of the source business object.

> **i Note**
>
> Condition evaluation checks the condition of the root node of a business object.

Condition Types

You can define the following conditions:

- Relevance Condition
  Based on the data of the source business object, the system checks whether the business object instance is relevant for service integration. The system evaluates the relevance condition first. If the result is true, the message is sent to the business object and the system evaluates the start condition. If the result is false, the message is not sent and the service integration process is not started.
- Start Condition
  This condition starts the service integration process, if the business object instance was not yet involved in the service integration process or if the process was canceled. If the result is true, the system starts the service integration process, that is, it compiles and sends the message to the business object.
- Change Condition
  If the service integration process for the business object instance has already started, the system evaluates the change condition. If the result is true, that is, the business object instance has changed, the

system sends a change message. If the result of the change condition is false, the system evaluates the cancel condition.

- Cancel Condition
  The system evaluates the cancel condition, if the result of the change condition is false.

Implementing the Evaluation Logic

When you create a condition, the system generates a script file. You implement the evaluation logic by using the scripting language. For more information, see Implement the Condition Evaluation Logic for Internal Communication [page 585].

The following methods are available for service integration script files:

- Return Statement
  The return statement is mandatory for each condition because it returns the result of the evaluation. The argument is a variable or a literal of Boolean type, for example, `return true;`. For more information, see Return Values (Business Logic) [page 269].
- `GetProcessContext` Method
  The `GetProcessContext` method is called for business object nodes. The following change notification fields are available for this method:
  - `IsActiveInProcess`
    The business object node was already sent and not canceled.

    > **Note**
    > You cannot use this field in a relevance condition.
  - `IsCreated`
    The business object node has been created.
  - `IsUpdated`
    The business object node has been updated.
  - `IsDeleted`
    The business object node has been deleted.

  For more information, see Process Context (Business Logic) [page 259].
- `InReconciliation` Parameter
  The `InReconciliation` parameter specifies that the service integration process is in reconciliation mode.
  For more information, see Script File Parameter (Business Logic) [page 270].

> **Example**
>
> The following example shows the code for the implementation of a relevance condition, a start condition and a change condition:
>
> - Relevance Condition
>
> ```
> return true;
> ```
>
> - Start Condition
>
> ```
> return(this.SalesProcessingStatusCode == "2" ||
>   this.SalesProcessingStatusCode == "3");
> ```
>
> - Change Condition
>
> ```
> return true;
> ```

The scenario for this implementation is as follows: Service integration is relevant if the sales process is started or finished. A start message will be sent if no message has been sent yet and the sales processing status is “In Process” (code value: 2) or “Finished” (code value: 3). A change message will be sent if the business object is changed and saved and the sales process is still in process or is finished. It is not necessary to implement a cancel condition because the change condition is always true.

8.14.3 Tasks

8.14.3.1 Configure Internal Communication

You can use internal communication to exchange data between business objects that are located in different deployment units. To communicate, the business objects exchange XML messages in an asynchronous way, which means that the source business object sends a request message to the target business object without expecting a response.

> Example
>
> You have defined the *Bonus Entitlement* business object in the *Customer Invoicing* deployment unit. The *Employee Compensation Agreement* business object is located in the *Human Capital Management* deployment unit. To transfer the bonus to the employees, you need to define an internal communication configuration between both business objects.

Prerequisite

You have created and activated your business object.

Procedure

1. Open your solution and, in the *Solution Explorer*, open the *Service Integration Wizard* either via the context menu of your business object or via the *Add New Item* dialog:

Solution Explorer: Business Object Context Menu

1. In the *Solution Explorer*, right-click a business object in your solution and select *Create Service Integration...*.
   The *Service Integration Wizard* opens.
2. In the *Select Service Integration Type* step, select *Internal Communication* and click *Next*.
3. In the *Create an Internal Communication* step, specify the sender and recipient of the communication by doing the following:
   1. Optionally, change the name and the sending business object .
   2. Select a receiving business object and a receiving business object node.

      > **i Note**
      >
      > One of the business objects must be a custom business object.
   3. Click *Next*

Solution Explorer: Add New Item Dialog

1. In the *Solution Explorer*, click *Add New Item* ( ).
   The *Add New Item* dialog opens.
2. Select *SAP* > *Service Integration* > *Internal Communication*, enter a name and click *Add*.
   The *Service Integration Wizard* opens.
3. In the *Create an Internal Communication* step, specify the sender and recipient of the communication by doing the following:
   1. Optionally, change the name. Select a sending business object, a receiving business object, and a receiving business object node.

      > **i Note**
      >
      > One of the business objects must be a custom business object.
   2. Click *Next*.

2. In the *Define Message Structure and Map to Receiving Business Object* step, specify the message definition and the identification attributes for the receiving business object as follows:
   1. Specify the structure of the message by selecting the elements of the receiving business object and its subnodes.
   2. Optionally, change the interaction message name. The default name is `Maintain`.
   3. Optionally, select a *Readable ID* and click *Next*.
3. In the *Map Message Structure to Sending Business Object* step, map the sending business object elements to the receiving business object elements. For example, map the `EmployeeID` element of the source business object *Bonus Entitlement* to the `EmployeeID` element of the `Item` node of the target business object *Employee Compensation Agreement*.
   To map an element, click a cell in the *Receiving Business Object* column. A dialog box appears in which you select the elements of the sending business object by double-clicking them. Only compatible fields are displayed.

   > **→ Tip**
   >
   > You can also use the *Propose Mapping* function.

   Click *Next*.
4. In the *Instance Identification for Update* step, define how node instances of the receiving business object are identified for update by doing the following:
   1. Under *Instance Identification*, select an alternative key or query.

- If you select *Alternative Key*, select the alternative key field.
- If you choose a query, for example, `QueryByElements`, choose the query parameters for the query mapping. Map each query parameter to a message field. Click the *Query Parameter* field twice and select a parameter, then click the *Message Field* field twice and select a field.

2. Under *Identification of Subnode Instance*, do the following:
   - To prevent data that is not part of the message from being deleted in the receiving business object, select *Prevent Data Deletion*.
   - Choose a subnode and the corresponding key field(s). If needed, you can choose additional subnodes and key fields.
3. Click *Next*.

5. In the *Review* step, check that the information you have entered is correct and click *Finish*.

   A new service integration definition with the file extension `.pid` ( ) appears in the *Solution Explorer*.
6. To activate your service integration settings, in the *Solution Explorer*, right-click the service integration definition and select *Activate*.

Related Information

Define Service Integration Error Handling [page 583]
Implement the Condition Evaluation Logic for Internal Communication [page 585]

8.14.3.2 Create XML File Input

In the SAP cloud solution, administrators can use an XML file to import third-party data from other systems into a business object.

To enable this functionality, you create an XML schema definition in the studio. This schema definition allows customers to create an XML file, which can then be used to import the data into the SAP cloud solution.

For more information, see the relevant documentation of the SAP cloud solution.

Prerequisite

You have created a solution.

Procedure

1. Open your solution and, in the *Solution Explorer*, open the *Service Integration Wizard* as follows:

Custom Business Object	SAP Business Object
1. In the *Solution Explorer*, right-click a business object in your solution and select *Create Service Integration*. The *Service Integration Wizard* opens. 2. In the *Select Service Integration Type* step, select *XML File Input* and click *Next*.	1. In the *Solution Explorer*, click *Add New Item* ( ). The *Add New Item* dialog opens. 2. Select *SAP* > *Service Integration* > *XML File Input*, enter a name and click *Add*. The *Service Integration Wizard* opens.

2. In the *Create an XML File Upload* step, do the following:
   1. Optionally, change the name. Select a receiving business object and a receiving business object node.
   2. Click *Next*.
3. In the *Define XML File Structure* step, do the following:
   1. Select the elements of the business object and its subnode.
   2. Optionally, select a *Readable ID* for identification in other processes.
   3. Optionally, select *List (Mass Processing)*. This allows your customer, for example, to upload several customer invoice requests with one `.xml` file.
   4. Click *Next*.
4. In the *Instance Identification for Update* step, define how node instances of the receiving business object are identified for update by doing the following:
   1. Under *Instance Identification*, select an alternative key or query.
      - If you select *By Alternative Key*, select the alternative key field.
      - If you select *By Business Object Query*, do the following:
        1. Select a query, for example, `QueryByElements`.
        2. Select the query parameters for the query mapping: Click the *Query Parameter* field twice and select a parameter, then click the *Message Field* field twice and select a field.
           Instead, you can also click *Propose Mapping*.
   2. Under *Identification of Subnode Instance*, do the following:
      - To prevent data that is not part of the message from being deleted in the receiving business object, select *Prevent Data Deletion*.
      - Choose a subnode and the corresponding key field(s). If needed, you can choose additional subnodes and key fields.
   3. Click *Next*.
5. In the *Review* step, check that the information you have entered is correct and click *Finish*.

   A new service integration definition with the file extension `.pid` ( ) appears in the *Solution Explorer*.
6. To activate your XML file input settings, in the *Solution Explorer*, right-click the service integration definition and select *Activate*.
   The system does the following:
   - It creates a folder within the standard WebDAV folder that is used by the SAP cloud solution. The name of the new folder is as follows: `...\<your project name>\<your XML file input name>`.
     Customers need to use this folder at a later point in time for their XML file.
   - It creates an XML schema definition.

   To download the XML schema definition to your local PC, proceed as follows:

7. In the *Solution Explorer*, double-click your service integration definition.
   The service integration definition opens in a document window and the information you have defined is displayed on different tabs. On the *Schema Definition* tab, you can review the XML schema definition.
8. Select the *General* tab. Click the schema definition link and save the XML schema definition to your local PC.
   You can now predefine the business tasks that an administrator can then monitor in the SAP cloud solution if errors or conflicts occur during the upload of third-party data. For more information, see Define Service Integration Error Handling [page 583].

> **i Note**
>
> If errors occur during the upload of third-party data to the SAP cloud solution, for example, when invalid data violates the XML schema definition, the error messages are displayed in the system in the *Application Log*. When error messages are raised by the business object itself they are displayed in the *Process Communication Errors* view for administrators, or tasks are created and assigned to the users responsible for the processing. The task are also displayed in the administrator's *Work* view within the *Application and User Management* work center.
>
> In cases of errors during the upload due to currency or alternate key inconsistencies in the XML file, the system displays the upload status under *Execution Details* with a red light. This indicates that the upload has failed. You can view the details of the error by clicking on the corresponding *Application Log ID*.
>
> Additionally, you can view the document under *Unprocessed Files* in the *Application and User Management* work center, *File Input* view. You can delete the file and upload the XML again after the required corrections have been made.

Result

Based on this XML schema definition, customers can create an XML file, for example, in Microsoft Office Excel. The XML file is to be stored in the WevDAV folder that was created by the studio.

8.14.3.3 Define Service Integration Error Handling

Overview

You can predefine the business tasks that an administrator can monitor in the cloud solution if errors or conflicts occur during asynchronous communication. To do this, you define the tasks and assign them to the appropriate messages of the target business object.

In the cloud solution, the tasks will be assigned to the administrator who is responsible for the support area that the target business object belongs to. To display the tasks, administrators navigate as follow:
Application and User Management > *Business Data Consistency* > *Process Communication Error* and then choose *Open Errors with Incidents* in the *Show* dropdown list.

Prerequisites

In the studio, you have created a service integration definition by creating an internal communication or xml file input using the *Service Integration Wizard*. For more information, see Configure Internal Communication [page 579] or Create XML File Input [page 581].

Procedure

1. In the studio, open the solution that contains the service integration definition for which you want to configure the error handling.
2. In the *Solution Explorer*, double-click the `*.pid` ( ) file.
   The service integration definition opens in a document window.
3. Select the *Error Handling* tab.
   On this tab on the left, the message structure of the target business object is displayed. On the right, you can define tasks that correspond to business object messages. These tasks are the business tasks that will appear in the cloud solution.
4. In the *Error Situations Solved by Task* area, enter a short text for a task, for example, `StartDateInvalid`.
5. Assign the appropriate target business object messages to your task. To do this, select a message in the message structure on the left and then click *Assign*. You can assign one or more messages to each task. The messages then appear in the *Assigned Messages* area.
6. In the *Error Situations Solved by Task* area, click *Long Text*.
   A text editor opens.
7. Enter the details about the error situation and how to solve it and then click *OK*.
   The text editor closes and your text is displayed in the *Long Text* field.
8. Save your service integration definition.
9. In the *Solution Explorer*, right-click the `*.pid` file and select *Activate*.
   During activation, a task list view with the name of the service integration definition and the `*.tasklistview` file extension is created. You can only display the task list view in the user interface designer (UI designer).
10. In the *Solution Explorer*, right-click a screen file and select *Open in UI Designer*.
11. In the UI designer, in the *Configuration Explorer*, open the *Process Communication Errors* work center view. The view is located in the following folder: `SAP_BYD_APPLICATION_UI/itsam/appmngt/pie/`. The file name of the view is `PCE_WCView`.
12. In the *Extensibility Explorer*, add the task list view to the *Process Communication Errors* work center view as follows:
    1. Select the `PCE_WCView.ao.uianchor` anchor and choose *Assign Component To View*.
    2. In the *Add Assigned Object* dialog box, select the `TaskListView` type.
    3. In the *Component* field, click ... and select your task list view. The view is located in the `SRC` subfolder of your project folder. For example, if the name of your project is `A2BCD300`, the folder name is `MAIN_A2BCD300/SRC/`.
    4. Click *Apply*.
13. Save and activate the `PCE_WCView` file.

Related Information

Service Integration Quick Guide [page 575]

8.14.3.4 Implement the Condition Evaluation Logic for Internal Communication

For internal communication, you can implement the business logic for condition evaluation to specify when a business object needs to send a message to the target business object, for example, a message to create or change a business object instance.

For more information, see Condition Evaluation for Internal Communication [page 577].

Prerequisites

In the studio, you have created a service integration definition by using the *Service Integration Wizard*. For more information, see Configure Internal Communication [page 579].

Procedure

1. In the studio, open the solution that contains the service integration definition for which you want to implement the condition evaluation.
2. In the *Solution Explorer*, open the service integration definition by double-clicking the `*.pid` file.
   The service integration definition opens in a document window.
3. On the *Conditions* tab, select a condition, for which you want to implement the evaluation logic, for example, `Relevance Condition`, and click *Add*.
   The studio generates a script file that opens in a separate document window.
4. In the code editor, define the evaluation logic for the condition.
   The condition must contain at least a *return* statement that returns the result of the condition evaluation with a Boolean return value. For more information, see Service Integration Methods [page 270].
5. Optional: Add further conditions and define the respective evaluation logic.
6. Save the service integration definition. Then right-click the service integration definition and select *Activate Script Files*.
   The script files for the implementation of the condition evaluation logic are activated.

> **i Note**
>
> The script files do not appear in the *Solution Explorer*. However, you can only select *Activate Script Files* if script files exist.

Related Information

Service Integration Quick Guide [page 575]

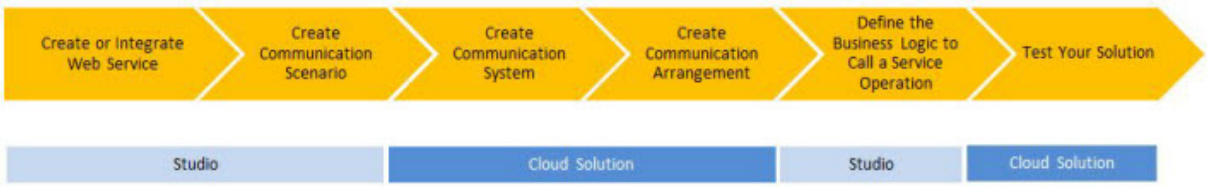
8.15 Web Services

8.15.1 Web Services Quick Guide

The studio allows you to create a Web service on an SAP business object or on any of your business objects, which can then be consumed by a client application. This enables remote access to the business object data. You can integrate an external Web service into your solution and address it in your business logic to enable access from the SAP cloud solution to a remote application. You can also select a set of inbound and outbound services to create a communication scenario. This allows customers to create communication arrangements in the SAP cloud solution to exchange business documents electronically. Communication for Web services is based on SOAP or REST.

Business and Technical Background

The following graphic shows you the steps involved in creating and integrating a Web service into your solution.



Web Services

A Web service is a set of programming standards that enables interoperation between different software applications, running on a variety of platforms or frameworks. Web services use Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Representational State Transfer (REST), and Web Services Description Language (WSDL) to integrate software applications over the internet. XML is used to tag the data, SOAP and REST are used to transfer the data, and WSDL is used to describe the Web service. Web service discovery mechanisms such as Universal Description, Discovery and Integration (UDDI), and Web Services Inspection Language (WSIL) can locate the Web services provided by a Web site.

For more information, see Web Services [page 591].

Web Service Operations

Web services contain multiple service operations, which are stateless synchronous inbound operations. The service operations are assigned to service interfaces which specify how to perform service requests and serve as collectors for specific operation types. There is one endpoint per service interface. The following service operations are available: `Read`, `Create`, `Update`, `Query`, and `Action`.

For more information, see Web Service Operations [page 596].

External Web Service Integration

You can integrate an external Web service into your solution to access a remote application using SOAP or REST. After creating a communication scenario and a communication arrangement, you can address the Web service in your business logic.

For more information, see External Web Service Integration [page 604].

Communication Scenarios

In a communication scenario, you can define a set of inbound and outbound services for a business process. Inbound services are Web services that you have created based on an SAP or custom business object, or Web services provided by SAP. Outbound services are external Web services that you have integrated into your solution. Based on a communication scenario, a customer can create a communication arrangement in SAP's cloud solution to exchange business documents electronically.

Communication Systems

A communication system represents an external system which is used for application integration. Communication systems can be, for example, external time recording applications or master data systems. You define a communication system in the SAP cloud solution and use it when you create the communication arrangement.

For more information, see the "Communication Systems Quick Guide" in the documentation of the SAP cloud solution.

Communication Arrangements

A communication arrangement helps you configure the electronic data exchange between your solution and the communication partner. It can be set up for multiple business documents and communication types. In the case of the communication type `B2B`, the communication partner is another company, for example a customer

or supplier; in the case of the communication type `Application Integration`, the communication partner is another system.

When you create a communication arrangement, you can specify an authentication method to authorize a user to call a Web service that you have created. You can create a communication arrangement for inbound and outbound services. An inbound service is a Web service that is based on a business object. An outbound service is a Web service that you have integrated into your solution.

For more information, see the "Communication Arrangements Quick Guide" in the documentation of the SAP cloud solution.

Tasks

Create a Web Service

You can create a Web service based on an SAP business object or one of your business objects. You select the business object elements and define the service operations that carry out tasks on the business object, for example, creating or updating the business object data.

For more information, see Create a Web Service [page 606].

Test a Web Service

Before you can use your Web service in a client application, you must test it in a separate testing tool.

For more information, see Test a Web Service [page 608].

Modify a Web Service

You can modify a Web service that you have created based on an SAP business object or one of your business objects. To modify a Web service, you can do the following:

- Add or remove business object elements for the service operations already defined in the Web service.
- Add or remove service operations.
- Adjust the derived names for the message data types used in the WSDL file.

For more information, see Modify a Web Service [page 609].

Edit Web Service Authorizations

You can edit the authorization of the business users assigned to work center views to use the Web service operations.

For more information, see Edit Web Service Authorizations [page 611].

Integrate an External Web Service Using SOAP

To enable access from the SAP cloud solution to an external application, you can create an external Web service integration scenario using SOAP

For more information, see Integrate an External Web Service Using SOAP [page 612].

Integrate an External Web Service Using REST

To enable access from the SAP cloud solution to an external application, you can create an external Web service integration scenario using REST.

For more information, see Integrate an External Web Service Using REST [page 614].

Example: Consume a REST Service

This example illustrates the consumption of a REST Web service in the business logic of a custom business object in the studio.

For more information, see Example: Consume a REST Service [page 616].

Modify an External Web Service Integration

You can modify an external Web service integration that you have created to access a remote application.

If you use SOAP-based communication, you can make the following changes:

- Replace the existing WSDL file with a new file
- Add or remove XML schema documents, if any are referenced in the WSDL file

For more information, see Modify an External Web Service Integration (SOAP) [page 621].

If you use REST-based communication, you can define or delete API keys used for authentication.

For more information, see Modify an External Web Service Integration (REST) [page 622].

Define a Communication Scenario

In a communication scenario, you can define a set of inbound and outbound services for a business process. Inbound services are Web services that you have created based on an SAP or custom business object, or Web

services provided by SAP. Outbound services are external Web services that you have integrated into your solution. Based on a communication scenario, a customer can create a communication arrangement in SAP's cloud solution to exchange business documents electronically.

For more information, see Define a Communication Scenario [page 619].

Modify a Communication Scenario Definition

You can modify a communication scenario that you have created based on a business-to-business or application integration communication type. You can modify a communication scenario as follows:

- Change the communication type on which the communication scenario is based.
- Add or remove inbound and outbound services.

For more information, see Modify a Communication Scenario [page 623].

Create a Communication System

To test the application integration scenario, you can create a communication system, which you can then use when you create the communication arrangement. When your customers use the solution in their production system, they create the communication system in the SAP cloud solution.

For more information, see Create a Communication System [page 624].

Create a Communication Arrangement

To test the communication scenario you have created, you need to create a communication arrangement in the SAP cloud solution.

For more information, see Create a Communication Arrangement [page 626].

Test a Service Operation

Before you can use your Web service in a client application, you must test it in a separate testing tool.

For more information, see Test a Service Operation [page 627].

Define the Business Logic to Call a Service Operation

If you have integrated an external Web service into your solution, you typically call the service operations provided by the Web service in a script file. The external Web service integration generates a library in the

project namespace. This library is similar to a reuse library and you can use it to implement your business logic to call the service operations.

Look for Deprecated Web Services in the SAP Cloud Solution

1. In the SAP cloud solution, in the *Application and User Management* work center, select the *Service Explorer* view.
2. Choose *Advanced* to open the advanced search pane.
3. Under *Operation* in the field *Release Status*, select **`Deprecated`** and in the field *Deprecation Release*, specify the release in which you want to search for deprecated Web services. Choose *Go*.

The system displays the list of deprecated Web services in the release you specified.

Further Information

Maintenance Mode

A customer-specific solution is in maintenance mode, if it has been assembled and downloaded; the solution status then is *Assembled*. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer's production tenant.

When your solution is in maintenance mode, please consider the following:

- You can add new fields to a service operation and delete them.
  In solution templates, you can also add new fields but you cannot delete them.
- You cannot delete delivered service operation fields.
- You can delete delivered service operation fields that have been deprecated.

8.15.2 Web Services

Overview

A Web service is a set of programming standards that enables interoperation between different software applications, running on a variety of platforms or frameworks. Web services use Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Representational State Transfer (REST), and Web Services Description Language (WSDL) to integrate software applications over the internet. XML is used to tag the data, SOAP and REST are used to transfer the data, and WSDL is used to describe the Web service. Web service discovery mechanisms such as Universal Description, Discovery and Integration (UDDI), and Web Services Inspection Language (WSIL) can locate the Web services provided by a Web site.

> **i Note**
>
> In the cloud solution, internal communication between business objects is implemented as service integration. For more information, see the Service Integration Quick Guide [page 575].

Features

Web Service Operations

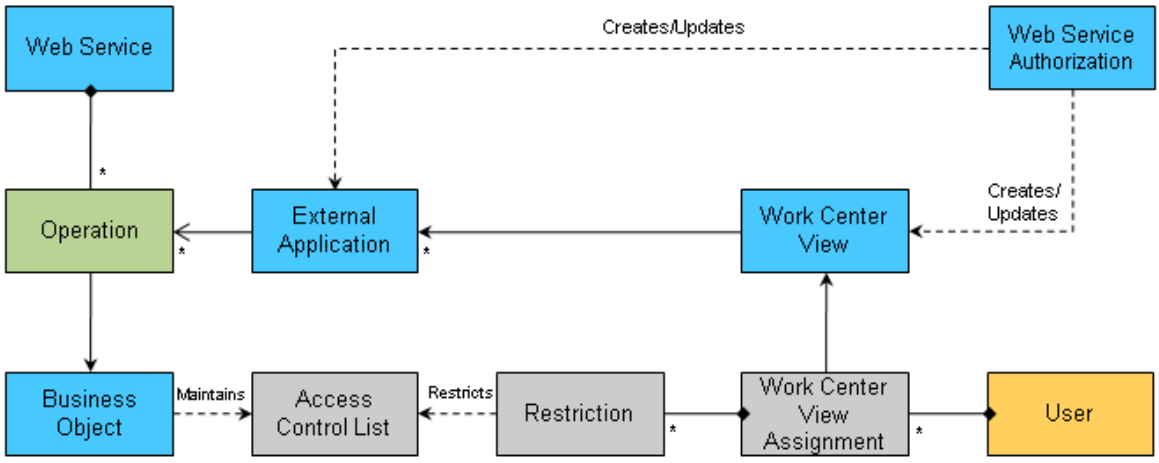
The function in the Web service is defined in service operations performed on specified elements in the selected business object.

For more information, see Web Service Operations [page 596].

Authorization Concept

You can authorize users to call a Web service that you have created as follows:

- **Web Service Authorization Object**
  You can define a Web service authorization object for the service operations in the Web service. The authorization concept is illustrated in the following diagram:



Web Service Authorization Process

  The Web service authorization object is assigned to work center views. For every assigned work center view, an external application containing the selected Web service operations is created. All users who have these work center views assigned to them are subsequently authorized to call the Web service operations. Any restrictions, such as read-only access to data, that have been defined for the assigned work center view are also taken into account when accessing business objects using the service operation.
  For more information about defining Web service authorizations, see Create a Web Service [page 606].
- **Communication Arrangement**
  You can add a Web service that you have created as an inbound service to a communication scenario. Based on this communication scenario, you can create a communication arrangement in the cloud

solution. In the communication arrangement, you can provide an authentication method to authorize a user to call the Web service.
For more information about communication arrangements, see the “Communication Arrangements Quick Guide” in the SAP cloud solution.

> **i Note**
>
> If you use a communication arrangement to authorize a Web service that has a Web service authorization object, you can delete the Web service authorization object.

Testing Consumption and Error Handling

To test the Web service, you download the WSDL file, import it into your client testing tool, and then assign the relevant work center views to a test user. For more information, see Test a Web Service [page 608].

The following list is intended to help you test the Web service:

- **Checking Endpoint Activation**
  Every studio tenant supports the WSIL standard. The WSIL service is a web site with the URL `https://[host name]//sap/bc/srt/wsil` that lists the active Web services and endpoints provided by SAP. Enter the URL for the WSIL service in your browser to check whether the endpoint is active and accessible.
- **XML Validation**
  As for all inbound service interfaces, the Web service runtime only performs a minimal set of checks (for example, checking maximum string lengths) and not general or strict validation of the inbound message. Superfluous tags are simply ignored.
- **Fault Messages**
  If an error occurs within a Web service, the `Log` element contains the details of the error in the response message. For more information about `Log` element, see Web Service Operations [page 596].

Web Services Description Language

The Web Services Description Language (WSDL) is an XML-based language that is used for describing the functions offered by a Web service. A WSDL file provides a description of how the Web service can be called, which parameters it expects, and which data structures it returns. WSDL is often used in combination with SOAP and an XML schema document to define Web services in the internet. A client application connecting to a Web service can read the WSDL file to determine which service operations are available on the server. The client application can then use SOAP to call one of the service operations listed in the WSDL file. Some of the data types used are defined in a separate XML schema document that is referenced in the WSDL file.

In a WSDL file the following elements are used for the definition of Web services:

- Types
  A container for data type definitions such as XML Schema Definition (XSD).
- Message
  An abstract, typed definition of the data being communicated.
- Operation
  An abstract description of an action supported by the service.

- Port Type
  An abstract set of operations supported by one or more endpoints.
- Binding
  A concrete protocol and data format specification for a particular port type. Optionally, a binding can be defined between the operation name and the SOAP action.
- Port
  A single endpoint defined as a combination of a binding and a network address.
- Service
  A collection of related endpoints.

A WSDL file defines Web services as collections of network endpoints, or ports. A port is defined by associating a network address with a reusable binding. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: messages and port types.

The following is an example of a basic WSDL file that refers to an XML Schema document:

Example

```
<?xml version=”1.0” encoding=”UTF-8”?>
<definitions targetNamespace="http://example.com/control/ws”>
  <types>
    <xsd:schema elementFormDefault="qualified" targetNamespace="http://
example.com/control" xmlns:ps="http://example.com/control">
      <xsd:import namespace="http://example.com/schema/control"
schemaLocation="PayrollControlSchema.xsd" xmlns:pcns="http://example.com/
schema/control"/>
      <xsd:element name="ProcessPayrollControl"
type="ps:ProcessPayrollControlType"/>
      <!— — Add more elements here — —>
    </xsd:schema>
  </types>
  <message name="GetAnnualPayrollSchedule">
  </message>
  <portType name="RequestPayrollControlServicesPortType">
    <operation name="ProcessPayrollControl">
    </operation>
  </portType>
  <binding name="RequestPayrollControlServicesBinding"
type="psws:RequestPayrollControlServicesPortType">
    <operation name="ProcessPayrollControl">
    </operation>
  </binding>
  <service name="PayrollControlServices">
    <port binding="psws:RequestPayrollControlServicesBinding"
name="RequestPayrollControlServicesPortType">
    </port>
  </service>
</definitions>
```

XML elements and attributes are usually placed inside a container called a namespace. Inside the `<types>` element, the `targetNamespace` attribute is used if you want to place elements and attributes from the default namespace that is defined using the `xmlns` attribute into a different namespace. The value of the `schemaLocation` attribute is the location of the XML schema document to use for the target namespace. Usually this value can be a URL or the location of the XML schema document relative to the location of the WSDL file on your local computer. In the above example, the WSDL file refers an XML schema document `PayrollControlSchema.xsd` that is located in the same folder as the WSDL file.

> **Note**
>
> The studio only accepts the relative location of the XML schema document for the value of the `schemaLocation` attribute.

XML Schema Document

An XML schema document describes and validates data in an XML document. It is written in the XML schema language, also referred to as XML Schema Definition (XSD).

Some of the data types used in a WSDL file are embedded in the file in the form of an XML schema document. This modular schema design maximizes the potential for reusability, flexibility, and maintainability of Web services components. As an XML schema document is written in XML, there is no need for a parser. An XML schema document defines a rich set of data types, for example, numeric data types, string data types, and date data types, and also makes it easier to validate documents based on namespaces.

An XML schema defines the following:

- Elements and attributes that can appear in a document.
- Whether an element is a child element.
- The order and the number of child elements.
- Whether an element is empty or can include text.
- Data types for elements and attributes.
- Default and fixed values for elements and attributes.

The following is an example of a basic XML schema document:

> **Example**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" targetNamespace="http://example.com/schema/
control">
  <xsd:import namespace="http://example.com/schema/common"
schemaLocation="PayrollCommonSchema.xsd"/>
  <xsd:element name="PayrollGroupAnnualSchedule" nillable="false"
type="pcns:PayrollGroupPayrollScheduleType"/>
  <xsd:complexType name="PayrollGroupPayrollScheduleType">
    <xsd:annotation>
      <xsd:documentation>Denotes the list of payroll schedules for the
calendar year for the payroll group</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="PayrollGroupCode" nillable="false"
type="cns:PayrollGroupCode"/>
      <xsd:element maxOccurs="unbounded" name="PayrollSchedule"
nillable="false" type="pcns:PayrollScheduleType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

In the above example, the XML schema document refers another XML schema document `PayrollCommonSchema.xsd`.

Related Information

Web Services Quick Guide [page 586]

8.15.3 Web Service Operations

Overview

Web services contain multiple service operations, which are stateless synchronous inbound operations. The service operations are assigned to service interfaces which specify how to perform service requests and serve as collectors for specific operation types. There is one endpoint per service interface. The following service operations are available: `Read`, `Create`, `Update`, `Query`, and `Action`.

Core Concepts

Statelessness

By definition and design, Web service operations are stateless synchronous inbound operations. This means that no resources such as enqueue locks or buffers are shared or maintained across subsequent calls. To ensure reasonable transaction sizes for mass operations, it is recommended that you avoid excessively large transactions because the time required to transfer huge amounts of data across the network may cause communication time-outs.

Action Code

The action code is a coded representation of an instruction to the recipient of a message telling the recipient how to process it.

Action Code List

Code	Name	Description
01	Create	The element is to be created at the recipient. The element must not exist at the recipient. The element ID and all data must be transferred.

Code	Name	Description
02	Change	The element is to be changed at the recipient. The element must exist at the recipient. The element ID and all data must be transferred.
03	Delete	The element is to be deleted at the recipient. The element must exist at the recipient. The element ID must be transferred. No data should be transferred, with the exception of elements that are mandatory due to their cardinality.
04	Save	The element is to be saved at the recipient. The element can exist at the recipient. If, at the recipient, the element: • Already exists, it is changed • Does not exist, it is created The element ID and all data must be transferred.
05	Remove	The element is to be deleted at the recipient. The element can exist at the recipient. If, at the recipient, the element: • Already exists, it is changed • Does not exist, it is created
06	No action	No action is to be carried out for the element at the recipient. The element must exist at the recipient. The element ID and all data must be transferred.

Complete Transmission Indicator

Due to the synchronous nature of Web services, the selective transmission of data during updates is essential for performance. In addition it cannot be assumed that the whole business object instance has to be read before an update takes place. This is especially true in cases where new field nodes are added via extensions. Without selective transmission, the new fields would be cleared and the new node instances would be deleted when an update message is processed. The `CompleteTransmissionIndicator` is used to indicate if the data in the requested operation represents all the data of an element or only part of it.

For operation message data types, the following rules apply:

- The `CompleteTransmissionIndicator` attribute is only used for lists, not for elements.
- For all updates, the default value is `false`, that is, by default, lists are always incomplete.
- For updates, if the value is `true`, all instances of the associated business object node that are not part of the message will be deleted.
- For creations or deletions, the indicator is assumed to be `true` or you can set it yourself explicitly to `true`.

Log

The `Log` element represents the result of an operation as a series of requests and responses.

The most important parts of the `Log` element are:

Description of Key Elements

Element	Details
`MaximumLogItemSeverityCode`, `LogItemSeverity`	These elements provide you with information on the degree of success of a given operation. The possible values are: • 1 Information • 2 Warning • 3 Error
`LogItem`	The aggregated data type for the`LogItem` contains the following elements: • In a response, the `ReferenceObjectNodeSenderTechnicalId` references the `ObjectNodeSenderTechnicalId` used in the request message. For more information, see the **Object Node Sender Technical ID** section below. • `ReferenceMessageElementName` This value denotes the element in the request that the log item refers to. It enables you to relate error messages to lines and columns in the query or request message, irrespective of the `ObjectNodeSenderTechnicalId`.

Object Node Sender Technical ID

When a request contains at least one element that occurs more than once, it must be possible to identify the corresponding elements in the response and the `LogItem`. To this end, the `ObjectNodeSenderTechnicalId` element is contained on every level of the request message as an optional element. The corresponding element `ReferenceObjectNodeSenderTechnicalId` is contained in the `LogItem` element at the root level of the response message.

The `ObjectNodeSenderTechnicalId` is also used to identify a failed newly created instance in a mass operation. Because the instance was not created in the database in these cases, no UUID or ID is returned and only the `ReferenceObjectNodeSenderTechnicalId` indicates which part of the request has failed.

The `ObjectNodeSenderTechnicalId` contains transient values that establish the correspondence between elements only for a single call.

Change State Identifier

Due to the stateless behavior of the Web service operations, no transactional or logical locking is possible. However, the `ChangeStateId` provides a way to ensure that a modifying operation is not executed if the state of the underlying business object instance in the database has changed since the last time that data was read. The `ChangeStateId` is an uninterpretable string that is provided by all operations and may be consumed by all modifying operations.

Update Operations

If the `ChangeStateId` is provided when calling an update operation, the system does not perform the operation if the state of the business object instance in the database has changed since the `ChangeStateId` was computed.

Create Operations

For create operations, there is no previous state of the underlying business object instance. Therefore the change state identifier is not modeled in single instance create operations and is optional in mass instance operations that support the corresponding action codes for creation.

If the change state identifier is specified and the specified value for a business object instance is different from the actual value computed from the database, an error message with the value`"SBM.SBM.CSM"` (change state mismatch) in the element `LogItem.LogItemCategoryCode` is returned for that business object instance in the `Log` element of the response.

> **i** Note
>
> Operations which create or update data, return the new IDs and the `ChangeStateId` only for those business object instances for which the operation succeeded.

Read Operation

The read operation returns detailed information of one or more instances of a business object. The part of the business object returned can be limited by the defined business object view. In general, the read operation always forms a pair with the corresponding create or update operation. Therefore, if a full or partial update operation for a business object exists, a full or partial read operation must also exist in order to retrieve the change state identifier for the update. A response message is always sent.

Result

The following is an example of a response message:

Example

```
<BusinessObject1ByIDResponse_sync xmlns:nm="http://sap.com/xi/SAPGlobal20/
Global">
  <BusinessObject1>
    <SAP_UUID>00163e01-27e0-1ed0-a6f9-129eeb25f05e</SAP_UUID>
    <id>1</id>
    <item>Washing Powder</item>
  </BusinessObject1>
  <Log />
</BusinessObject1ByIDResponse_sync>
```

Note

Note that response and confirmation messages may contain an XML namespace definition for `xmlns:prx="urn:sap.com:proxy…"`. This is only used for support purposes and should ignored at runtime.

Create Operation

The create operation creates a business object instance. A confirmation message is always sent.

Result

The following is an example of a confirmation message:

Example

```
<BusinessObject1viewCreateConfirmation_sync xmlns:nm="http://sap.com/xi/
SAPGlobal20/Global">
  <BusinessObject1>
    <SAP_UUID>00163e01-27e0-1ed0-a6f9-a08d04afd1f8</SAP_UUID>
    <id>3</id>
  </BusinessObject1>
  <Log>
    <MaximumLogItemSeverityCode>1</MaximumLogItemSeverityCode>
    <Item>
      <TypeId>007(/CSG_RUNTIME/)</TypeId>
      <SeverityCode>1</SeverityCode>
      <Note>Create operation was successful</Note>
    </Item>
  </Log>
</BusinessObject1viewCreateConfirmation_sync>
```

Update Operation

The update operation updates a single instance of an object with checks for concurrent updates. The part of the business object susceptible to the update can be limited by the defined business object view. The update operation may add and/or remove parts of the business object but it is not meant to be used for status changes. A confirmation message is always sent.

The update operation checks for concurrent updates. It checks the `ChangeStateId` element to determine whether somebody else has changed the same business object instance since the last read. If this is the case, then an error message is returned.

Result

The following is an example of a response message:

Example

```
<nm:BusinessObject1viewUpdateConfirmation_sync xmlns:nm="http://sap.com/xi/
SAPGlobal20/Global">
  <Log>
    <MaximumLogItemSeverityCode>1</MaximumLogItemSeverityCode>
    <Item>
      <TypeId>013(/CSG_RUNTIME/)</TypeId>
      <SeverityCode>1</SeverityCode>
      <Note>Update operation was successful</Note>
    </Item>
  </Log>
</nm:BusinessObject1viewUpdateConfirmation_sync>
```

Query Operation

When defining a query operation in the *Web Service Creation Wizard*, you can only use queries that already exist for the business object. The query operation retrieves specified information in business object instances. Query operations only read data; there is no change of any persistent data. A response message is always sent. The basic query naming pattern is:

```
Find[View]SimpleBy[Selection criteria]
```

Where:

- The business object view defines the response structure.
- The selection criteria view splits the operation so that each operation offers a semantically meaningful set of selection criteria typically needed for a given purpose. It is determined from the name of the underlying business object query.

Maximum Number of Rows Returned

You can define the maximum number of rows returned by a query using the following elements of the `ProcessingConditions` element:

- `QueryHitsUnlimitedIndicator`
- `QueryHitsMaximumNumberValue`

If the `QueryHitsUnlimitedIndicator` is not set and the `QueryHitsMaximumNumberValue` is zero, the default of 100 rows is assumed.

⚠ Caution

The caller of the Web service is responsible for creating a reasonable package size. Therefore, we do not recommend that you set the unlimited indicator to true or that you return more than 1000 rows.

Note that the `LastReturnedObjectID` element is not supported in query response messages. Queries return a filtered result and may offer sorting criteria. Therefore, choosing a key or ID will depend on the query parameters, the returned fields, and the sorting order used in the query results. Also, not all business object nodes will have a key or the key will not have a linear order, for example if it is a UUID. The caller has to decide which fields can be used in each specific case. In simple cases the ID element of the node for which the query is defined is sufficient. In complex cases, all elements of the response have to be treated as part of the key.

Comparisons Permitted

You can use the following types of comparisons in Web service query operations:

- Equals

  > ⚠ Caution
  >
  > If an Equals string contains an asterisk (*), the system implicitly interprets this as a wildcard query and thus as a CONTAINS pattern. You cannot therefore query items that have the asterisk in their name. For example, `QUERY PRODUCT WHERE PRODUCT ID=A*` returns all products beginning with the letter A. It does not return the details of the product name A*

- Between
- Less than
- Less than or equal to
- Greater than
- Greater than or equal to

Each query parameter includes a `SelectionBy[Name]`, where the name determines the query parameter, and has the following structure:

Details of Query Parameters

Element	Value	Description
InclusionExclusionCode	I	Include in the result set
	E	Exclude from the result set
IntervalBoundaryTypeCode	1	Equal to Single Value; = X
	2	Between (excluding upper boundary) Interval with closed lower and open upper boundary; [X, Y)
	3	Between Interval with open lower and open upper boundary; (X, Y)
	4	Between (excluding both boundaries) Interval with open lower and open upper boundary; (X, Y)

Element	Value	Description
	5	Between (excluding lower boundary) Interval with open lower and closed upper boundary; (X, Y]
	6	Less than Interval with unlimited lower and open upper boundary; < X
	7	Less Than or Equal to Interval with unlimited lower and closed upper boundary; <= X
	8	Greater Than Interval with open lower and unlimited upper boundary; > X
	9	Greater Than or Equal to Interval with closed lower and unlimited upper boundary; >= X

Result

The following is an example of a response message:

Example

```
<nm:BusinessObject1QueryByElementsSimpleByConfirmation_sync xmlns:nm="http://
sap.com/xi/SAPGlobal20/Global">
  <BusinessObject1>
    <SAP_UUID>00163e01-3318-1ed0-b4df-43192913f03f</SAP_UUID>
    <ID>1</ID>
    <FirstName>Abc</FirstName>
    <MiddleName>D</MiddleName>
    <LastName>Xyz</LastName>
    <State>1</State>
  </BusinessObject1>
  <ProcessingConditions>
    <ReturnedQueryHitsNumberValue>1</ReturnedQueryHitsNumberValue>
    <MoreHitsAvailableIndicator>false</MoreHitsAvailableIndicator>
  </ProcessingConditions>
  <Log>
    <MaximumLogItemSeverityCode>1</MaximumLogItemSeverityCode>
    <Item>
      <TypeID>009(/CSG_RUNTIME/)</TypeID>
      <SeverityCode>1</SeverityCode>
      <Note>Query returned 1 records</Note>
    </Item>
  </Log>
</nm:BusinessObject1QueryByElementsSimpleByConfirmation_sync>
```

Action Operation

When defining an action operation in the *Web Service Creation Wizard*, you can only use actions that already exist for the business object. The action operation changes the state of a business object instance. It is not intended to be used to modify data. The request message types are typically very short; they often contain just the business object ID. A response message is always sent.

Result

The following example shows an action response message:

Example

```
<nm:BusinessObject1ResetStateResetStateConfirmation_sync xmlns:nm="http://
sap.com/xi/SAPGlobal20/Global"">
  <Log>
    <MaximumLogItemSeverityCode>1</MaximumLogItemSeverityCode>
    <Item>
      <TypeId>017(/CSG_RUNTIME/)</TypeId>
      <SeverityCode>1</SeverityCode>
      <Note>Action RESETSTATE992EBFB4F6ADFACD executed</Note>
    </Item>
  </Log>
</nm:BusinessObject1ResetStateResetStateConfirmation_sync>
```

Related Information

Web Services Quick Guide [page 586]

8.15.4 External Web Service Integration

You can integrate an external Web service into your solution to access an external application using the SOAP or REST protocol.

What is SOAP?

SOAP (Simple Object Access Protocol) is an xml-based protocol specification for exchanging structured information via Web services. The integrated Web service is described by a WSDL file. With SOAP, you can use different transport protocols, for example, HTTP or SMTP. SOAP uses named operations, each implementing business logic through different interfaces. WS-Security and Reliable Messaging are supported.

For more information, see Integrate an External Web Service Using SOAP [page 612].

What is REST?

REST (Representational State Transfer) is an architectural style for designing networked applications. REST services provide a uniform interface using standard HTTP operations (such as GET, POST, PUT...) and exchange representations of resources. JSON and XML are frequently used formats in REST services.

For more information, see Integrate an External Web Service Using REST [page 614].

To integrate REST Web services, you can also use a mashup. For more information, see Create a Mashup Web Service [page 559].

What is the difference between SOAP and REST?

The SOAP and REST protocols both provide a set of rules for requesting information from a server. The following table lists some of the differences. It should also help you decide which protocol to use for your integration scenario. Note also that Web services often support either REST or SOAP.

Category	SOAP	REST
Handling	complex	simple
Format	XML	XML, CSV, JSON, RSS
Protocol	HTTP, SMTP	HTTP, HTTPS
Means used to define Web service	WSDL	URL
Purpose	for APIs	for simple transactions, CRUD operations, fast results

How do I integrate an external Web service into my solution?

The following graphic shows what you need to do to integrate an external Web service into your solution:

External Web Service Integration

The procedures for integrating SOAP and REST Web Services are very similar. However, the steps you need to take in the *External Web Service Wizard* are quite different as you can see in the following table:

SOAP	REST
1. Upload a WSDL file. 2. Optional: Select XML schema document. 3. Decide if you want to use stateful communication. For more information, see Integrate an External Web Service Using SOAP [page 612].	1. Enter URL of Web service that you want to use. 2. Optional: Allow non-secure communication or define API keys. For more information, see Integrate an External Web Service Using REST [page 614].

Related Information

Example: Consume a REST Service [page 616]

8.15.5 Tasks

8.15.5.1 Create a Web Service

You can create a Web service based on an SAP business object or one of your business objects. You select the business object elements and define the service operations that carry out tasks on the business object, for example, creating or updating the business object data.

The Web service can then be consumed by a client application to access the business object data remotely.

Prerequisites

- If you create a Web service based on an SAP business object, the business object must be released for the SAP public solution model (PSM).
- If you create a Web service based on your business object, the business object must be activated.

Procedure

1. In the *Solution Explorer*, do one of the following:
   - If you create a Web service based on an SAP business object or any of your business objects, proceed as follows:
     1. Right-click the project in your solution and select *Add* > *New Item*.
     2. Select *SAP* > *Service Integration* > *Web Service*, enter a name and click *Add*.
        The *Web Service Creation Wizard* opens.
   - If you create a Web service based on one of your business objects, right-click the business object and select *Create Service Integration...*.
     1. The *Service Integration Wizard* opens.
     2. Select *Web Service* and click *Next*.
2. In the *Enter Web Service Name* step, proceed as follows:
   1. If you used the *Add* > *New Item* function, optionally, change the name and select a namespace and the business object based on which you want to create the Web service.
   2. If the Web service is based on your business object, enter a name for the Web service.
   3. Click *Next*.
3. In the *Select Elements* step, select the business object elements that you need for the Web service operations. If you have already created a Web service for the business object, you must enter a business object view name.
   If you create a Web service based on a business object that has an association to another business object, you can also add elements of the associated business object. For more information, see Association (Business Object) [page 196].

   > **i Note**
   >
   > If you have included a dependent object in your business object definition, you can only select the dependent object node and not the elements. The elements of the dependent object, however, are

included. For more information on dependent objects, see Dependent Object (Business Object) [page 203].

Click *Next*.

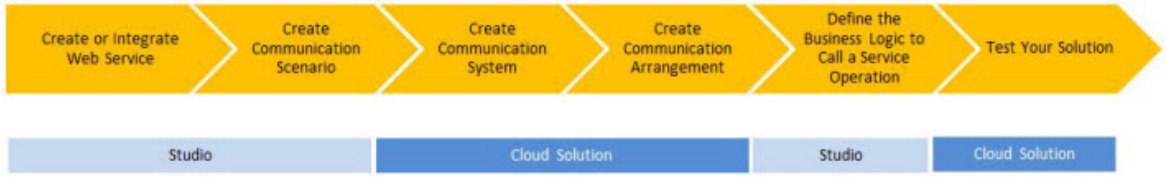
4. In the following steps, define the service operations for the Web service. For each service operation that you define, select the checkbox, enter a name, and select on which business object elements you want to perform the service operation.
   The following Web service operations are provided by the studio:
   - **Create** service operation
   - **Read** service operation
   - **Update** service operation
   - **Query** service operation:
     In the *Define a Query Service Operation* step, enter a service operation name and select a query.

     > **i Note**
     >
     > If you define a query service operation with query parameters that are not released for the PSM, you cannot activate the Web service.

   - **Action** service operation:
     In the *Define a Action Service Operation* step, enter a service operation name and select an action.

   For more information about service operations, see Web Service Operations [page 596].
   Click *Next*.
5. In the *Define Web Service Authorization* step, assign your service operations to work center views. This will authorize business users who are assigned to the views to use the service operations in the Web service.
   Proceed as follows:
   1. Click *Add*.
   2. In the *Add Work Center View* dialog, do one of the following:
      - Click *Use Existing View* to select a view.
      - Click *Create New View* and enter a name.
        The view will be created when you have completed all steps in the wizard.
   3. Click *Next*.
   4. For each work center view, select the service operations that you want to assign.

   For more information about the authorization concept, see Web Services [page 591] under *Authorization Concept*.
6. Review your settings and click *Finish*.
   A Web service with the file extension `.webservice`
   (



) appears in the *Solution Explorer*. Below the Web service, the Web service authorization object with the extension `.wsauth` () is listed, which contains the authorization settings of the Web service.
7. In the *Solution Explorer*, right-click the Web service and select *Activate*.

Result

- **WSDL File**
  The Web service and the Web service authorization object are activated. You can now download the WSDL file, which can be consumed by a client application. To download the file, right-click the Web service and select *Download WSDL File*.
- **Adjust Derived Names**
  You need to adjust the derived names in the WSDL file if you are prompted to do so. You can also optionally choose to adjust the names for ease of use. For more information, see Modify a Web Service [page 609].
- **Work Center View**
  If you entered a name for a new work center view while defining the Web service authorization settings, the view appears in the *Solution Explorer* with the file extension `.uiwocview` ( ). To edit the work center view, right-click the file and select *Open in UI Designer*. For more information, see Model a Work Center View [page 710].
- **Maintenance Mode**
  You can create a Web service when your solution is in maintenance mode. For more information, see Maintenance Mode [page 185].

Related Information

Test a Web Service [page 608]

8.15.5.2 Test a Web Service

Overview

Before you can use your Web service in a client application, you must test it in a separate testing tool.

Prerequisites

- You have created and activated the Web service and have defined the Web service authorizations.
  For more information, see Create a Web Service [page 606].
- If you want to test a Web service, it must be in scope.

  If you have created BAC elements in your solution and assigned your solution content to a business option, you can perform scoping manually in the development environment of the SAP cloud solution. You must be assigned the Business User role and have access rights for the *Business Configuration* work center. For more information, see User Setup Quick Guide [page 33].

  Alternatively, you can trigger the deployment of business configuration in the *Solution Explorer* by right-clicking the solution and selecting *Deploy Business Configuration*.

> **i Note**
>
> If you have defined BAC elements for your solution, a dialog box opens when you use the *Deploy Business Configuration* function. You have to select one of the following options:
>
> - Deploy all business configuration content for your solution
> - Deploy only the business configuration content for your solution that is assigned to the business options you have selected in the business adaptation catalog
>
> This allows you to test both partial and complete activation of your business configuration content in the development environment.

Procedure

1. In the studio, in the *Solution Explorer*, right-click the Web service that you want to test and select *Download WSDL File*.

   > **i Note**
   >
   > If an error occurs when you download the WSDL file, check that your user has been created according to the procedure described in the User Setup Quick Guide [page 33].

2. Import the WSDL file into your client testing tool.
3. In the SAP cloud solution, assign a test user to the work center views authorized for the Web service. Note that the work center views that are authorized to access the Web service must have been assigned to a work center. You can then assign test users to this work center in the system for the SAP cloud solution. To do so, navigate to the *User and Access Management* work center, *Business Users* view.

   > **⚠ Caution**
   >
   > This step needs to be repeated when your customers go live with your final solution. You have to ensure that any users who should have access to the Web service are assigned to the relevant work center views in the customer's productive system.

4. Test the Web service.
   For more information about the syntax used, see Web Service Operations [page 596].

8.15.5.3 Modify a Web Service

You can modify a Web service that you have created based on an SAP business object or one of your business objects. To modify a Web service, you can do the following:

- Add or remove business object elements for the service operations already defined in the Web service.
- Add or remove service operations.
- Adjust the derived names for the message data types used in the WSDL file.

Prerequisite

You have created a Web service based on an SAP business object or one of your business objects. For more information, see Create a Web Service [page 606].

Procedure

1. In the *Solution Explorer*, double-click the `.webservice` file of the Web service that you want to modify.
2. To modify your Web service, do one of the following:
   - To add or remove a business object element available for use in the Web service, select the *Business Object View* tab. Select or deselect the element that you want to add or remove.
     If you deselect a business object element, it will be removed from all existing service operations. If you add an element, it will become available for use in all existing service operations.

     > **i Note**
     >
     > If you have included a dependent object in your business object definition, you can only select the dependent object node and not the elements. The elements of the dependent object, however, are included. For more information on dependent objects, see Dependent Object (Business Object) [page 203].

   - To add or remove a business object element from a specific service operation, select the *Service Operations* tab. Proceed as follows:
     1. From the dropdown list, select the service operation from which you want to add or remove a business object element.
     2. Select or deselect the relevant business object element.
   - To add or delete a service operation, select the *Service Operations* tab.
     - To add a service operation to your Web service, proceed as follows:
       1. Click *Add*.
          The *Add Service Operation* dialog box opens.
       2. Select a service operation and enter a unique name for the service operation. Click *Add*.
     - To delete a service operation from your Web service, proceed as follows:
       1. From the dropdown list, select the service operation that you want to delete.
       2. Click *Delete*.
   - To adjust the derived name for a message data type used in the WSDL file, select the *Message Data Types* tab.
     When you create a Web service, the corresponding request and response messages for each service operation contain derived names based on the names of the business object elements and Web service operations. You may be prompted to adjust these derived names if they exceed the 120 characters limit. You can also optionally choose to adjust the names for ease of use.
     To adjust the derived name, proceed as follows:
     1. Select a service operation and then choose either the request or response message type.
        The derived names used in the message are displayed.
     2. Select a line in the structure and enter an alternative name in the *Abbreviated Name* field.
     3. Click *Apply*.
        The new name is displayed in the message structure. If you renamed a line in the structure that has subitems, then the subitems are renamed accordingly.

If you want to reset a derived name of a message type to its initial name, select the service operation and choose the request or response message type. Click *Reset Tree to Initial Names*.

3. Click *Save*.
4. In the *Solution Explorer*, right-click the Web service and select *Activate*.

> **i Note**
>
> When you re-activate an existing Web service in the studio, an updated version of the Web service's WSDL file is generated. You must download the new WSDL file after you re-activate the Web service.

Result

The modified Web service is activated. The system updates the Web service authorization object with the changes you made to existing service operations. However, if you have added new service operations, you must edit the authorizations manually. For more information, see Web Services Quick Guide [page 586] under *Edit Web Service Authorizations*.

You can now download the updated WSDL file, which then can be consumed by a client application to access the business object data remotely.

8.15.5.4 Edit Web Service Authorizations

You can edit the authorization of the business users assigned to work center views to use the Web service operations.

Prerequisites

You have opened a solution that contains a Web service. For more information, see Create a Web Service [page 606].

Procedure

1. In the *Solution Explorer*, double-click the `.wsauth` file to open it for editing.
2. You have the following options:
   - To add a new work center view, click *Add*, select *Create New View*, and enter a technical name for a new view.
   - To remove authorization from a work center view, select the line in the table and click *Remove*. Note that this does not delete any work center views created specifically for the Web service.
   - If you have removed authorization from a work center view, you can add it as follows: Click *Add*, select *Use Existing View*, select the view, and click *Add*.

- To assign the Web service operations to work center views, select a view and then select or deselect the checkboxes in the lower table.
- To find out where a work center view is stored in the UI designer, view the *Repository Path* column.

3. Save your changes and activate the Web service.

> **i Note**
>
> If you delete a Web service authorization object in the *Solution Explorer*, any work center views that you created during Web service creation are not deleted. You can delete these manually. You can recreate a deleted Web service authorization by right-clicking the Web service and selecting *Create Web Service Authorization*.

8.15.5.5 Integrate an External Web Service Using SOAP

To enable access from the SAP cloud solution to a remote application, you can create an external Web service integration scenario using SOAP.

SOAP (Simple Object Access Protocol) is an xml-based protocol specification for exchanging structured information via Web services. The integrated Web service is described by a WSDL file. With SOAP, you can use different transport protocols, for example, HTTP or SMTP. SOAP uses named operations, each implementing business logic through different interfaces. WS-Security and Reliable Messaging are supported.

For more information, see External Web Service Integration [page 604].

Prerequisites

- You have opened a solution in the studio.
- You have saved the WSDL file of the Web service that you want to integrate into your solution on your local computer.
- If the WSDL file refers to one or more XML schema documents, you have saved the schema documents on your local computer.
- If the WSDL file refers to XML schema documents and the value of the `schemaLocation` attribute in the WSDL file is a URL, you have replaced the URL with the location of the XML schema document on your computer. Note that the document's location must be relative to the location of the WSDL file.
  For more information about WSDL and XML schema document, see Web Services [page 591].

> **i Note**
>
> Save the WSDL file and the XML schema documents in the same location on your computer. In this case, the values of the `schemaLocation` attribute are the XML schema document names only.

Procedure

1. In the *Solution Explorer*, right-click your solution and select *Add New Item*.
   The *Add New Item* dialog opens.
2. Select *SAP* > *Service Integration* > *External Web Service Integration* and click *Add*.
   The *External Web Service Integration Wizard* opens.
3. In the *Enter Web Service Name and Type* step, enter a name for the Web service and select *SOAP*. Click *Next*.
4. In the *Import WSDL File Documents* step, do the following:
   1. Select the WSDL file of the Web service that you want to integrate into your solution.
   2. Select the *Use Session ID* checkbox, if you want the external Web service to allow stateful communication, that is, return a session ID/cookie when it is called.
   3. Optional: If the WSDL file refers to one or more XML schema documents, click *Add*, select the XML schema document and click *Open*.
      Change the port name, if required.
   4. Click *Next*.
5. In the *Define Communication Scenario* step, do the following:
   1. Optional: Select the *Create Communication Scenario* checkbox to create a communication scenario that includes the Web service integration that you have defined.
      If you choose not to select this option, you must assign the Web service integration to a communication scenario later. For more information, see Define a Communication Scenario [page 619].
   2. Enter a name for the communication scenario. It must start with a capital letter and can have 25 characters (letters, digits, and underscores).
   3. Optional: Select the *Allow Non-Secure URLs (HTTP)* checkbox to allow non-secure communication.
   4. Click *Next*.
6. On the *Review* page, review your entries and click *Finish*.

   In the *Solution Explorer*, an external Web service integration file with the extension `.wsid` ( ) together with the `.wsdl` file ( ) appears. If you have created a communication scenario, the corresponding `.csd` file ( ) is also displayed.
7. In the *Solution Explorer*, right-click the `.wsid` file and click *Activate*.
   If you have created a communication scenario, right-click the `.csd` file and click *Activate*.

Result

You have created an external Web service integration and can now do the following:

- **Edit your external Web service integration.**
  To do so, right-click the `.wsdl` file. The document window opens where you can edit the Web service integration definition
  For more information, see Modify an External Web Service Integration (SOAP) [page 621].
- **Test the service operations**.
  Right-click the `.wsid` file and select *Test Service Operations*.

For more information, see Test a Service Operation [page 627].

- **Create a communication scenario**
  If you have not created a communication scenario yet, right-click the `.wsid` file and select *Create Communication Scenario Definition*.
  For more information, see Define a Communication Scenario [page 619].
- **Create a communication system**
  If you have created a communication scenario, right-click the `.csd` file and select *Manage Communication System*. Afterwards, you can use the communication system to create a communication arrangement.
  For more information, see Create a Communication System [page 624] and Create a Communication Arrangement [page 626].

8.15.5.6 Integrate an External Web Service Using REST

To enable access from the SAP cloud solution to an external application, you can create an external Web service integration scenario using REST.

REST (Representational State Transfer) is an architectural style for designing networked applications. REST services provide a uniform interface using standard HTTP operations (such as GET, POST, PUT...) and exchange representations of resources. JSON and XML are frequently used formats in REST services.

For more information, see External Web Service Integration [page 604].

If you want to learn how you can use this function in a real-life example, see Example: Consume a REST Service [page 616].

Prerequisites

You have opened a solution in the studio.

Procedure

1. In the *Solution Explorer*, right-click your solution and select *Add New Item*.
   The *Add New Item* dialog opens.
2. Select *SAP* > *Service Integration* > *External Web Service Integration* and click *Add*.
   The *External Web Service Integration Wizard* opens.
3. In the *Enter Web Service Name and Type* step, enter a name for the Web service and select *REST*. Click *Next*.
4. In the *Enter End Point of REST Service* step, do the following:
   1. Enter the URL of the service.
      The *Host Address*, *Port*, and *Path* fields are filled automatically.
   2. Click *Next*.

5. In the *Define Communication Scenario* step, do the following:
    1. Optional: Select the *Create Communication Scenario* checkbox to create a communication scenario that includes the Web service integration that you have defined.
    If you choose not to select this option, you must assign the Web service integration to a communication scenario later. For more information, see Define a Communication Scenario [page 619].
    2. Enter a name for the communication scenario. It must start with a capital letter and can have 25 characters (letters, digits, and underscores).
    3. Optional: Select the *Allow Non-Secure URLs (HTTP)* checkbox to allow non-secure communication.
    4. If the service you are calling requires authentication, you can define the required identifiers as API keys. The values of the API keys are maintained in the communication arrangement. You cannot use API keys with non-secure URLs (HTTP). To define an API key, do the following:
        1. Enter the key in the *API Key* field.
        2. Click the *Placeholder for API Key in Script File* field to the right of the *API Key* field.
        The placeholder is generated, which you can use in the script file to build the Web service request.
    5. Click *Next*.
6. On the *Review* page, review your entries and click *Finish*.

    In the *Solution Explorer*, an external Web service integration file with the extension `.wsid` ( ) is displayed. If you have chosen to create a communication scenario, a communication scenario definition file with the extension `.csd` ( ) is also displayed.
7. In the *Solution Explorer*, right-click the `.wsid` file and click *Activate*.
    A communication scenario and a communication system are generated in the SAP cloud solution. These can be used to create a communication arrangement.

Result

You have created an external Web service integration and a communication scenario and can now do the following:

- **Edit your external Web service integration.**
    To do so, right-click the `.wsid` file and select *Open*. The document window opens where you can edit the Web service integration definition.
    For more information, see Modify an External Web Service Integration (REST) [page 622].
- **Create a communication scenario**
    If you have not already created a communication scenario, right-click the `.wsid` file and select *Create Communication Scenario Definition*.
    For more information, see Define a Communication Scenario [page 619].
- **Edit a communication scenario**
    To edit your communication scenario, double-click the `.csd` file to open it.
    For more information, see Modify a Communication Scenario [page 623].
- **Create a communication arrangement.**
    To do so, right-click the `.csd` file and select *Manage Communication Arrangement.*
    For more information, see Create a Communication Arrangement [page 626].
- **Define the business logic to call a service operation.**
    For more information, see Define the Business Logic for a Business Object [page 428].

When you create the script file, you can use the following functions provided in the SAP reuse library:

- WebServiceUtilities. ExecuteRESTService
- WebServiceUtilities.EncryptWithHashBasedMessageAuthenticationCode

For more information, see Web Service Utilities [page 307].

8.15.5.7 Example: Consume a REST Service

Overview

This example illustrates the consumption of a REST Web service in the business logic of a custom business object in the studio.

> ### Example
>
> **Use case**
>
> In a Web application, users enter their country and postal code on a screen. The REST service automatically fills the city field.

To test this example, please use a REST Web service that displays the city when you enter the corresponding postal code. You can find such REST Web services free of charge on the Internet.

Here is what you do in this example:

Example: Consume a REST Service

Prerequisites

- You have a partner developer user in the studio and a business user in the SAP cloud solution. For more information, see User Setup Quick Guide [page 33].
- You have created a solution in the studio. For more information, see Create a Solution [page 177].

Define an external Web service integration scenario

1. In the studio, in the *Solution Explorer*, open your solution and click *Add New Item* ( ).
2. In the *Add New Item* dialog, select *External Web Service Integration*, enter the name `GetCityByPostalCode` name and click *Add*.
   The *External Web Service Integration Wizard* opens.
3. In the *Enter Web Service Name and Type*, select the web service type *REST*, and click *Next*.
4. In the *Enter End Point of REST Service* step, enter the *URL* of the REST Web service and click *Next*.

5. In the *Define Communication Scenario* step, do the following:
    1. Select *Create Communication Scenario*, and enter **`GetCity`** as the *Communication Scenario Name*.
       If the Web service can only be called using HTTP instead of HTTPS, select *Allow Non-Secure URLs (HTTP)*.
    2. Click *Next*.
6. In the *Review* step, review your entries and press *Finish*.
   A `.csd` file and a `.wsid` file are added to the *Solution Explorer*.
7. Right-click the `.wsid` file and click *Activate*.
8. Right-click the `.csd` file and click *Activate*.

Create a communication arrangement

1. In the *Solution Explorer*, right-click the `.csd` file and select *Communication Arrangement*.
   The *Communication Arrangements* view opens.
2. Click *New*.
3. In the *New Communication Arrangement* view, do the following:
    1. In the *Select Scenario* step, select the `GetCity` communication scenario and click *Next*.
    2. In the *Define Business Data* step, enter the *System Instance ID* GETCITY-[]namespace of your solution]_ and click *Next*.
    3. In the *Define Technical Data* step, click *Edit Advanced Settings*, and do the following:
        1. Under *Details: GetCity*, deselect the *Use Basic Settings* checkbox.
        2. In the *Application Protocol* field, select **`Http`** and, if required, change the protocol to **`Hypertext Transfer Protocol (http)`**.
        3. In the *Authentication Method* field, select *None*.
        4. Click *Check Completeness* and make sure that no errors occur, then click *Next*.
    4. In the *Review* step, review your entries, and click *Finish* and *Close*.

Create a business object

1. In the studio, in the *Solution Explorer*, open your solution and click *Add New Item* ( ).
2. In the *Add New Item* dialog, select *Business Object*, enter the name **`Address`** and click *Add*.
3. In the editor, enter the following code:

```
import AP.Common.GDT as apCommonGDT;
businessobject Address {
 [Label ("Country")] element CountryCode: CountryCode;
[Label ("Postal Code")] element PostalCode:PostalCode;
[Label ("City")] element City: LANGUAGEINDEPENDENT_LONG_Name;
}
```

   Save and activate the business object.

Create a script file

1. In the *Solution Explorer*, right-click the business object and select *Create Script Files*.
2. In the *Create Script Files* dialog, select **`Event: After Modify`** and click *OK*.
   An `.absl` file is generated.
3. Enter the following code:

```
import ABSL;
// Communication details
var ScenarioName = "GetCity";
var ServiceName = "GetCityByPostalCode";
var HttpMethod = "GET";
var HttpResource = ""; // not required for this example
var ContentType = ""; // not required for this example
var Body = ""; // not required for this example
var HeaderParameter : collectionof NameAndValue;  // not required for this
example
```

```
// Set URL Parameter
var URLParameter    : collectionof NameAndValue;
var URLParameterEntry : NameAndValue;
URLParameterEntry.Name  = "country";
URLParameterEntry.Value = this.CountryCode;
URLParameter.Add(URLParameterEntry);
URLParameterEntry.Name  = "postalcode";
URLParameterEntry.Value = this.PostalCode;
URLParameter.Add(URLParameterEntry);
URLParameterEntry.Name  = "username";
URLParameterEntry.Value = // Add your user name;
URLParameter.Add(URLParameterEntry);
```

```
// Execute webservice call
var ws_result = WebServiceUtilities.ExecuteRESTService(ScenarioName,
ServiceName, HttpMethod, HttpResource,  URLParameter, HeaderParameter,
ContentType, Body);
```

```
//Parse result of Web service and retrieve city name; you may have to adapt
the logic to the Web service used
var position = ws_result.Content.Find("placeName");
if ( position != -1)
{ var start = position + 12;
position = ws_result.Content.Find("lat");
if ( position != -1)
{ var length = position - start - 3;
this.City = ws_result.Content.Substring(start, length);
}
else { this.City = ""; };
}
else { this.City = ""; };
```

4. Save and activate the script file.

Create screens

1. In the *Solution Explorer*, select the business object and select *Create Screens*.
2. In the *Create Screens* dialog, select a screen scenario, enter a *Short ID* and click *OK*.
   Several `.uicomponent` files are generated.

Test the Web service

1. Right-click the `OWL.uicomponent` file and select *Preview Screen*.
2. Log on to the SAP cloud solution.
3. In the object work list, click *New*.
4. In the quick activity, select a country and enter a postal code of your choice, then click *Save*.
   The city field is automatically filled by the REST Web service.

The REST Web service has filled the City field

8.15.5.8 Define a Communication Scenario

In a communication scenario, you can define a set of inbound and outbound services for a business process. Inbound services are Web services that you have created based on an SAP or custom business object, or Web services provided by SAP. Outbound services are external Web services that you have integrated into your solution. Based on a communication scenario, a customer can create a communication arrangement in SAP's cloud solution to exchange business documents electronically.

For more information about communication arrangements, see the "Communication Arrangements Quick Guide" in the documentation of the SAP cloud solution.

Prerequisites

- You have opened a solution in the studio.
- You have checked the available inbound services in the *Service Explorer* of the SAP cloud solution. The *Service Explorer* lists all released inbound services available for the solution, along with service documentation and technical data. For more information, see the "Service Explorer Quick Guide" in the documentation of the SAP cloud solution.
- You have created and activated an inbound service or an outbound service.

  > **i Note**
  >
  > An inbound service is a Web service that is based on a business object. An outbound service is a Web service that you have integrated into your solution.

  For more information about creating an inbound Web service, see Create a Web Service [page 606].
  For more information about integrating an outbound Web service, see Integrate an External Web Service Using SOAP [page 612] and Integrate an External Web Service Using REST [page 614].

> **i Note**
>
> There are three ways to define a communication scenario:
>
> 1. You can use the *Add New Item* dialog. This procedure is described below.
> 2. You can use the *Define Communication Scenario* step of the *External Web Service Integration Wizard*.
> 3. After integrating an external Web service using SOAP or REST into your solution, you can right-click the `.wsid` file in the *Solution Explorer* and select *Create Communication Scenario Definition*.

Procedure

1. Right-click your solution and select *Add* > *New Item*.
2. Select the *Communication Scenario* template, enter a name, and click *Add*.
   The *Communication Scenario Definition Wizard* opens.
3. In the *Enter Communication Scenario Name* step, do the following:
   1. Enter a name for the communication scenario and select the Web service type *SOAP* or *REST*.
      For more information, see Integrate an External Web Service Using SOAP [page 612] and Integrate an External Web Service Using REST [page 614].
   2. **SOAP** only: Select the communication type:
      - *Business-to Business (B2B)* to define the electronic exchange of business documents with a business partner.
      - *Application Integration (A2X)* to define the electronic exchange of business documents with a communication system, for example, an external time recording system.

      For more information about communication types, see the “Communication Arrangements Quick Guide” in the documentation of the SAP cloud solution.
   3. Optional: Select *Allow Non-Secure URLs (HTTP)*.
   4. Click *Next*.
4. In the *Select Services and Operations* step, select the inbound and outbound services that you want to include in your communication scenario and click *Next*.
5. **REST** only: In the *Enter API Keys for Communication Scenario* step, do the following:
   1. In the *API Key* field, enter the name of the API key that is relevant for the external Web service. Your customer can then enter values for the API key in the corresponding communication arrangements in the cloud solution. For security reasons, you cannot access the values of the API key in your business logic. However, if you want to use the values, you can use the generated placeholders instead.
   2. Click the *Placeholder for API Key in Script File* field to the right of the *API Key* field.
      The placeholder is generated, which you can use in the script file.
   3. Click *Next*.
6. Review your settings and click *Finish*.

   A communication scenario definition with the file extension `.csd` ( ) appears in the *Solution Explorer*.
7. In the *Solution Explorer*, right-click the communication scenario definition and select *Activate*.

Result

You have created a communication scenario definition in the studio. You can now do the following:

- **Create a communication system in the SAP cloud solution**
  If you have chosen to create an application integration scenario, you can now create a communication system.
  For more information, see Create a Communication System [page 624].
- **Create a communication arrangement in the SAP cloud solution**
  For more information, see Create a Communication Arrangement [page 626].
- Optional: **Test the execution of the service operations** provided by the external Web service that you have integrated into your solution.
  For more information, see Test a Service Operation [page 627].
- **Call the service operations of the inbound and outbound services in a script file.**
  For more information, see Define the Business Logic for a Business Object [page 428].

8.15.5.9 Modify an External Web Service Integration (SOAP)

You can modify an external Web service integration that you have created to access a remote application.

If you use SOAP-based communication, you can make the following changes:

- Replace the existing WSDL file with a new file
- Add or remove XML schema documents, if any are referenced in the WSDL file

Prerequisites

- You have created an external Web service integration.
  For more information, see Integrate an External Web Service Using SOAP [page 612].
- If you want to replace the current WSDL file with a new file, you have saved the new WSDL file on your local computer.
- If the WSDL file refers to one or more XML schema documents, you have saved the schema documents on your local computer.

Procedure

1. In the *Solution Explorer*, double-click the `.wsid` file ( ) of the external Web service integration that you want to modify.
2. To modify your external Web service integration, select the *General* tab. Do one of the following:
   - To import a new WSDL file, click *Upload WSDL File* and select the file from the location on your computer where you have saved it.
   - If the WSDL file refers to one or more XML schema documents, select the schema document from your local computer and click *Add*.

- If you want to remove one or more XML schema documents that you have already added, select the schema document from the list and click *Remove Document*.

> **i Note**
>
> If you import a new WSDL file and XML schema documents were uploaded for the previous WSDL file, the schema documents are deleted on saving the changes.

3. To view the service operations provided by the external Web service, select the *Service Operations* tab.
4. Click *Save*.
5. In the *Solution Explorer*, right-click the external Web service integration file, and select *Activate*.

Result

- To test the execution of the service operations provided by the external Web service, right-click the `.wsid` file and click *Test Service Operations*.
  For more information, see Test a Service Operation [page 627].
- The external Web service integration is re-activated. You can add it as an outbound service in a communication scenario and create a communication arrangement based on the communication scenario. For more information, see Define a Communication Scenario [page 619] and Create a Communication Arrangement [page 626].

Related Information

Modify an External Web Service Integration (REST) [page 622]

8.15.5.10 Modify an External Web Service Integration (REST)

You can modify an external Web service integration that you have created to access a remote application.

If you use REST-based communication, you can define or delete API keys used for authentication.

Prerequisites

You have created an external Web service integration. For more information, see Integrate an External Web Service Using REST [page 614].

Procedure

1. In the *Solution Explorer*, double-click the `.wsid` file ( ) of the external Web service integration that you want to modify.
2. To modify your external Web service integration, do one of the following:
   - On the *General* tab, select the *Allow Non-Secure URLs (HTTP)* checkbox.

     > **i Note**
     >
     > If you select this checkbox, you cannot use API keys any longer.

   - On the *API Keys* tab, delete or define API keys.
3. Click *Save*.
4. In the *Solution Explorer*, right-click the external Web service integration file, and select *Activate*.

Result

The external Web service integration is re-activated. You can add it as an outbound service in a communication scenario and create a communication arrangement based on the communication scenario. For more information, see Define a Communication Scenario [page 619] and Create a Communication Arrangement [page 626].

Related Information

Modify an External Web Service Integration (SOAP) [page 621]

8.15.5.11 Modify a Communication Scenario

You can modify a communication scenario that you have created based on a business-to-business or application integration communication type. You can modify a communication scenario as follows:

- Change the communication type on which the communication scenario is based.
- Add or remove inbound and outbound services.

Prerequisites

You have defined a communication scenario. For more information, see Define a Communication Scenario [page 619].

Procedure

1. In the *Solution Explorer*, double-click the `.csd` file of the communication scenario that you want to modify.
2. To modify your communication scenario, do one of the following:
   - Change the communication type on which the communication scenario is based.
   - Select or deselect the inbound and outbound services that you want to add or remove.
3. Click *Save*.
4. In the *Solution Explorer*, right-click the communication scenario and select *Activate*.

Result

The communication scenario definition is re-activated. You can now use the modified communication scenario to create a communication arrangement in the SAP cloud solution. For more information, see Create a Communication Arrangement [page 626].

8.15.5.12 Create a Communication System

To test the application integration scenario, you can create a communication system, which you can then use when you create the communication arrangement. When your customers use the solution in their production system, they create the communication system in the SAP cloud solution.

A communication system represents an external system which is used for application integration. Communication systems can be, for example, external time recording applications or master data systems. You define a communication system in the SAP cloud solution and use it when you create the communication arrangement.

For more information, see the “Communication Systems Quick Guide” in the documentation of the SAP cloud solution.

Prerequisites

- You have defined an external Web service integration using SOAP or REST as described in the following table:

You have defined an external Web service integration scenario using SOAP	You have defined an external Web service integration scenario using REST
In the *External Web Service Integration Wizard*, in the *Enter Web Service Name and Type* step, you have chosen to define an *Application Integration* scenario.	The communication system is created automatically. However, you can follow the procedure described below to change the communication system or its properties.
For more information, see Integrate an External Web Service Using SOAP [page 612].	For more information, see Integrate an External Web Service Using REST [page 614].

- You have created a communication scenario. For more information, see Define a Communication Scenario [page 619].

Procedure

1. In the studio, in the *Solution Explorer*, right-click the communication scenario definition (`.csd`) file and select *Manage Communication System*.
2. The SAP cloud solution opens. Log on to the system.
   The *Communication Systems* view opens.
3. Click *New*.
   The *New Communication System* screen opens.
4. Under *Basic Information*, enter an *ID* for the communication system, the *Host Name*, and the *System Access Type*.

   > → Tip
   >
   > You can find the host name in the `.wsdl` file.

5. Click *Save and Close*.
6. Refresh the screen.
   The communication system that you have created is displayed in the *Communication Systems* view.
7. Select your communication system and click *Actions* > *Set to Active*.
   The status of the communication system changes to *Active*.

Result

You have created a communication system. You can now use it to create a communication arrangement to test the application integration scenario. For more information, see Create a Communication Arrangement [page 626].

8.15.5.13 Create a Communication Arrangement

To test the communication scenario you have created, you need to create a communication arrangement in the SAP cloud solution.

Usually, your customer sets up the communication arrangement for your solution in the SAP cloud solution. However, to test the communication scenario you have created, you need to create a communication arrangement as well.

For more information on communication arrangements, see the “Communication Arrangements Quick Guide” in the documentation of the SAP cloud solution.

Prerequisites

You have created a communication scenario in your solution.

Procedure

1. In the studio, in the *Solution Explorer*, right-click the communication scenario definition (`.csd`,) file and click *Manage Communication Arrangement*.
2. Log on to the SAP cloud solution.
   The *Communication Arrangements* work center view opens.
3. Click *New*.
4. The *New Communication Arrangement* view opens.
5. In the *Select Scenario* step, select the communication scenario that you have created in the studio and click *Next*.

   > **i** Note
   >
   > When you hover over the name of the communication scenario, its key is displayed. It contains the namespace of your solution.
   >
   > When there is a solution with a communication scenario, and a patch solution is also created for the solution, you will see two communication scenarios with the same name in the system. In such cases, if you wish to create a communication arrangement to test the communication scenario within the patch solution, then you must select the communication scenario with the word **patch** in brackets.

6. In the *Define Business Data* step, select the business partner and click *Next*.
7. In the *Define Technical Data* step, do the following:
   1. Select the *Application Protocol* and *Authentication Method*. You can select one of the following options:
      - *SSL Client Certificate*: Enter a certificate.
      - *User and Password*: Click *Edit Credentials* and, in the *Authentication* dialog, enter a *User ID* and, optionally, a password.
      - *None* (REST only)

2. Enter the *Host Name*.
   You can find it in your solution in the studio: In the *Solution Explorer*, open the `.wsid` file, go to the *General tab*, and copy the name from the *Host Address* field.
3. Click *Next*.

8. In the *Review* step, review your entries and click *Check Completeness*. Then click *Finish* and *Close*.

Result

Your communication arrangement is created. You can now test your web service.

For more information, see Test a Web Service [page 608].

8.15.5.14 Test a Service Operation

You can use the Test Tool to test the execution of the service operations provided by an external Web service.

Prerequisites

- You have integrated an external Web service using SOAP and activated the `.wsid` file.
  For more information, see Integrate an External Web Service Using SOAP [page 612].
- You have created a communication scenario.
  For more information, see Define a Communication Scenario [page 619].
- A communication arrangement exists for the communication scenario.
  For more information, see Create a Communication Arrangement [page 626].

Procedure

1. In the *Solution Explorer*, right-click the `.wsid` file of the external Web service integration for which you want to test the execution of service operations and select *Test Service Operations*.
   The *Test Tool* opens.

   > **i Note**
   >
   > The *Test Tool* tab is not available in the following cases:
   >
   > - You cleaned the `.wsid` file.
   > - You deleted the `.wsdl` file.
   > - You uploaded a new `.wsdl` file to replace the original file.

2. Select a service operation.
3. Provide test values for the request message in the *Request* tab and click *Submit Request*.

4. Select the communication scenario and click *OK*. If the communication scenario is based on a B2B communication type, enter the business partner name.
5. Click *OK*.
   You can see the response message in the *Response* tab and verify if it is correct.

Result

You have tested the integrated Web service. If you are not satisfied with the result, you can edit the external Web service integration. For more information, see Modify an External Web Service Integration (SOAP) [page 621].

8.16 Report an Incident

If you encounter a problem during development, you can report an incident to request help in solving the problem. the SAP cloud solution collects context data, such as system information and relevant business data, and attaches it to the incident. This gives information about the system at the time at which the incident occurred and helps SAP Support to better understand the problem. Note that the context data is only useful if you report the incident from the screen on which the problem occurred, as it includes information that is specific to that screen. This means it is important that you always report your solution-related incidents from the screen on which they occur in your system.

You report a problem that is not related to your UI components in the *Partner Development* work center in the SAP cloud solution. For example, you create an incident in this work center if you encounter issues with the studio or the user interface designer.

> **i** Note
>
> You cannot create an incident from the studio or the user interface designer.

Prerequisites

If you create the incident for SAP Support from the SAP cloud solution, you must be logged on to that system with your **development user**. Development users have administrator rights in the SAP cloud solution and are assigned to the *Incidents* view. This view is available in the SAP cloud solution as follows:

- In the SAP Cloud for Customer solution: *Administrator* > *Incidents*
- In all other SAP cloud solutions: *Application and User Management* > *Incidents*

Note that you can also create an incident with a standard business user. However if that business user is not an administrator in the SAP cloud solution, the system forwards the incident to the responsible development user with administrator rights, who in turn forwards it to SAP Support.

Procedure

1. In the SAP cloud solution, click *Help* > *Report Incident*.
2. Enter the required details.

8.17 Debugging

8.17.1 Debugging Quick Guide

You can use debugging to observe the runtime behavior of your solution and to locate logic errors.

For information about how to access the debug functions, see Debugging User Interface [page 635].

> **i** Note
>
> Each time you start the debugger, tracing is also started automatically. For more information about tracing, see here [page 641].

The main tool that you use for debugging is the debugger, which works with your script files and allows you to break the execution of your solution at specified breakpoints to examine your code and to display values of variables defined in your solution.

For more information, see About Debugging [page 631].

Tasks

Debugging a Solution

This simple process explains how you can easily debug your solution. For more information, see here [page 638].

Set a Breakpoint

1. In the *Solution Explorer*, open a script file.
   The script file opens in the code editor.
2. You can set a breakpoint in the code editor by doing one of the following:
   - In the left grey margin of the code editor, click a line of executable code.
   - Click a line of executable code and press **F9**.

   The glyph in the margin indicates that a breakpoint is set.

Delete a Breakpoint

To delete breakpoints in your current script file, proceed as follows:

- To delete a particular breakpoint, click or press **F9** in the code line, where the breakpoint has been set.
  The margin is cleared and the red glyph is deleted.
- To delete all breakpoints in your script file, in the toolbar of the *Breakpoints* tool window, click *Delete all breakpoints*

Disable or Enable a Breakpoint

- To disable breakpoints that are set in the current script file, proceed as follows:
  - To disable a particular breakpoint, in the *Breakpoint* tool window, clear the checkbox of the enabled breakpoint.
    The breakpoint is now disabled.
    To disable all breakpoints in your script file, in the toolbar of the *Breakpoints* tool window, click *Enable or disable all breakpoints*
- To enable breakpoints that have been disabled before, proceed as follows:
  - To enable a particular breakpoint, in the *Breakpoint* tool window, select the checkbox of the disabled breakpoint.
    The breakpoint is enabled again.
  - To enable all disabled breakpoints, in the toolbar of the *Breakpoints* tool window, click *Enable or disable all breakpoints*

Start and Stop Debugging

- To start debugging, in the *Debug* toolbar, click or press **F5**.
  The code editor is brought to the foreground and *Debug* windows are enabled, for example, the *Breakpoints* window and the *Locals* window. The debugger is in run mode and you cannot edit your code anymore. You can now open a screen, for example, by using the preview function, to debug your solution.
- To stop debugging, in the *Debug* toolbar, click or press **SHIFT** + **F5**.
  The *Debug* windows are closed and you are in design mode again, that is, you can now edit and change your code.

Step Through the Code

In break mode, you can perform the following activities to step through your code:

Your Activity	Icon	Ksey
Step through the code one statement at a time		`F10`
Hit the next breakpoint		`F5`
You are inside a script file and want to return to the calling script file		`SHIFT` + `F11`

→ Tip

If you have inspected another script file and want to return to the script file, where the running process has stopped, in the *Debug* toolbar, click .

8.17.2 Business and Technical Background

8.17.2.1 About Debugging

Debugging is an important process during the development of a solution which enables you to observe the runtime behavior of your solution and to locate logic errors. The main tool that you use for debugging is the debugger, which works with your script files and allows you to break the execution of your solution at specified breakpoints to examine your code and to display values of variables defined in your solution.

Each time you start debugging in the studio, tracing is also started automatically. The tracing information enriches the debug information with informational messages and, if applicable, with error messages. For more information, see Tracing [page 641].

Debugging Basics

Debug Modes

During design mode, which is the mode when you are implementing the business logic for your solution, you cannot use the debugger. However, you can manage breakpoints, for example, set or delete them.

After you have started debugging, you cannot edit or change your code anymore. The following debug modes are available:

- Run Mode

This mode is divided into the following phases:

  - The debugger is waiting to attach to a process that is represented, for example, by the actions that you perform on a screen in your solution. In this mode, you can manage breakpoints, for example, set or delete breakpoints.
  - The debugger is already attached to a process and is executing the business logic. This phase ends when execution is completed or when the debugger hits a breakpoint.
    During this phase you can set breakpoints, but the breakpoints are only enabled after execution is completed.

- Break Mode
  You have opened a screen in your solution, for example, by using the preview function in the studio. When you perform actions on this screen and a break occurs, your solution and the debugger are in break mode. During break mode, you can use all debugging functions and inspect the information provided by the debugger.

Breakpoints

Breakpoints enable you to suspend execution where and when you need to. You can allow your solution to run until it hits a breakpoint, and then you can start to debug your code.

The following functions are available for breakpoints:

- Setting and Deleting Breakpoints
  You set or delete a breakpoint in the left margin of your script file manually. A breakpoint location is indicated by a glyph in the margin. All breakpoints that are set in your solution are listed in the *Breakpoints* window.
- Disabling and Enabling Breakpoints
  You can disable or enable a breakpoint that is already set. The glyph indicates that the breakpoint is enabled. The glyph indicates that the breakpoint is disabled. If you disable a breakpoint, you can still see where it was set, but the debugger does not stop execution at this position anymore. Only if you enable the breakpoint again, the debugger will stop execution when it hits this breakpoint.
- Exporting and Importing Breakpoints
  You can export breakpoints to a local XML file, for example, to create a backup or to share them with another user.

For information about how to set and manage breakpoints, see Debugging Quick Guide [page 629].

Stepping

One of the common debugging procedures is stepping, which means, executing code one statement at a time. Execution stops if the debugger reaches one of the following:

- A code line that contains an executable statement.
  If a code line contains more than one executable statement, the debugger executes them one statement at a time.
- An enabled breakpoint.
- The end of the script file.

If the debugger reaches the end of a script file, it steps out of the current script file and stops at the next executable statement in the calling script file. Execution continues if there is no call from another script file.

Frequently Asked Questions

Why can't I set a breakpoint in this code line?

Please check the following:

- **Does the current code line contain an executable statement?**
  If not, you can't set a breakpoint in this code line.
- **Has the script file been activated?**
  If not, stop debugging and then activate the script file. Restart debugging and then set the breakpoint.
- **Did you already reach a number of 30 breakpoints in your script file?**
  You have reached the maximum number of breakpoints in a script file. Before you are able to set a new breakpoint, delete one or more breakpoints which you don't need.

Why didn't the debugger hit a specific breakpoint?

Please check the following:

- **Has debugging been started and if yes, has it been started for the correct user?**
  In the *Options* dialog box, on the *Tracing and Debugging* tab, check whether the setting for the business user is correct. If you want to enable debugging for your development user, please make sure that the *Enable for business user* checkbox is deselected. For more information about this setting, see Debug User Interface [page 635] under *Debug Settings* > *Debugging and tracing: Enable for business user*.
- **Does the breakpoint look like this:** ○?
  If yes, the breakpoint is disabled. Please enable the breakpoint by selecting the checkbox of the disabled breakpoint in the *Breakpoint* tool window. Now the breakpoint is enabled and looks like this: ●.
- **Did you activate the script file or the solution after you have set the breakpoint?**
  If not, the breakpoint is not valid. Please stop debugging, close the screen, and then activate the solution. Restart debugging and open the screen again, for example, by using the preview function.
- **Did you open the screen before you have activated the script file or the solution?**
  If yes, close the screen and open it again, for example, by using the preview function.
- **Did the debugger already hit a breakpoint in the other session?**
  Please keep in mind that the debugger evaluates only one session at a time. A session typically corresponds to the actions performed on one screen. Although you are able to perform actions in parallel on another screen during the debug run, you can't debug the code of these actions.
- **Is the debugger still executing the business logic in another session?**

  In the *Debug* toolbar, you have clicked or and the debugger is still executing the business logic in the other session. Please wait until execution is finished. You can then start execution in the next session.

When I am stepping through the code, why can't I see some variable values?

The value of a variable doesn't show in the *Locals* window, when the following applies:

- The variable is currently not in scope, that is, it is defined in a code block of a *foreach* or *if* statement.
- The variable contains a query. As the compiler optimizes the code, query variables are not available at runtime.
- The value is not stored in a variable. The value is the result of an association or a function, which is part of a path expression, but which is not explicitly assigned to a variable.

Can I step within a path expression?

No, stepping executes the entire statement.

Can I debug more than one session in parallel at a time?

No, the debugger evaluates only one session at a time. A session typically corresponds to the actions performed on one screen. Although you are able to perform actions in parallel on another screen during the debug run, you can't debug the code of these actions.

Why doesn't the Output window update the tracing information after a certain period of time?

If you have opened your screen, for example, by using the preview function, and you haven't performed any actions on this screen for 10 minutes, tracing will stop automatically.

If you want to enable tracing again, restart debugging and perform the actions on the screen again.

Can I enable debugging to examine actions performed by a specific business user?

Yes, you can. For more information, see Debug User Interface [page 635] under *Debug Settings* > *Debugging and tracing: Enable for business user*.

Can I cancel a session using the debugger?

No, this is not possible. Just close the screen on which you are performing the actions. This will cancel the debug session.

How can I debug an internal communication scenario?

In an internal communication scenario, business objects exchange XML messages in an asynchronous way, which means that the source business object (source BO) sends a request message to the target business object (target BO) and the target BO processes this message accordingly. The processes related to the source BO and the processes related to the target BO run in different sessions.

The debugger can stop either in the script files of the source BO or in the script files of the target BO. If you want to debug the script files of the target BO, please make sure that the breakpoints in the script files of the source BO are disabled.

Can I debug a Web Service in the Cloud Applications Studio?

No. Debugging of a Web Service is not possible since a technical user is required for the execution. This technical user is different from the user that executes that web service.

Which user can I use to debug an XML file input run?

Debug the actions with the business user who schedules the file input run. For this, in the *Options and Settings* dialog box on the *Debugging and Tracing* tab, enter the name of the relevant business user. For more information about this setting, see Debug User Interface [page 635] under *Debug Settings* > *Debugging and tracing: Enable for business user*.

8.17.2.2 Debugging User Interface

The studio's debugging user interface is composed of several elements, which enable you, for example, to select debugging functions, to view information provided by the debugger, and to specify debug settings.

Debug Menu

You can find all debugging and tracing functions in the *Debug* menu. There you can also enable the appropriate windows, for example, the *Call Stack* window or the *Output* window.

Debug Toolbar

The *Debug* toolbar provides buttons that allow you to start and stop debugging together with tracing and also to start and stop tracing separately. If you have already started debugging, the tracing buttons are inactive. The *Debug* toolbar is automatically enabled when you start debugging. To enable the *Debug* toolbar separately, choose *View* > *Toolbars* > *Debug*.

Debug Windows

The debug windows display information the debugger provides and support you in analyzing this information. The following windows are provided:

- Locals
- Breakpoints
- Call Stack
- Output

To enable a debug window, choose *Debug* > *Windows* and select the appropriate window.

> **i** Note
>
> You can only enable the *Locals* window and the *Call Stack* window if debugging has started.

Locals Window

This window displays variables local to the current script file. You can view the name, the value, and the type of the variable. If the variable type is a structure, a business object node or a collection, you can expand or collapse the information provided. You cannot edit this information.

Breakpoints Window

This window lists all breakpoints that are set in your solution. The *Breakpoints* window also contains a set of functions, which allow you, for example, to disable and to enable breakpoints and to export breakpoints to an XML file. For more information about breakpoints, see About Debugging [page 631].

Output Window

In the *Debug* view of the *Output* window you can view tracing information that the system is collecting during the debug run. Each script file is listed in the order it was called and can be nested if it was called from another script fie.

The information displayed in the *Debug* view is the same information that you can find in the *Trace Explorer* window. However, the information in the *Debug* view is displayed during the debugging process whereas the information in the *Trace Explorer* window is displayed after debugging or tracing has been stopped. The format in the *Debug* view is also quite different, that is, each line starts with a time stamp and the session number and severity, for example, `[1:E]` for session 1, containing an error message in the specified script file.

> **i** Note
>
> The debugger evaluates one session in parallel at a time. A session typically corresponds to the actions performed on one screen. Although you are able to perform actions in parallel on another screen during the debug run, you cannot debug the code of these actions.

Call Stack Window

The *Call Stack* window supports you in finding out which script file called the script file that is currently processed.

The arrow identifies the script file which is currently processed. If you double-click in the window, the corresponding code line in the script file is highlighted in yellow. The arrow identifies the script file from which the script file with the arrow has been called. If you double-click in the window, the corresponding script file is brought to the foreground and the code line containing the call is highlighted in green.

DataTips

A DataTip is a box in the code editor that displays the value of a variable in your solution similar to the *Locals* window. When the debugger is in break mode, you can enable a DataTip in the current script file by placing the mouse pointer over a variable in the code editor.

You can pin a DataTip to a specific location in the script file or you can leave it floating on top of all studio windows. For more information about DataTips, see the Microsoft Visual Studio documentation at http://www.microsoft.com .

> **i** Note
>
> DataTips are not supported if the path to the element includes an association or a function call.

Debug Settings

You can set debug settings in the *Options* dialog box as a default. To change the debug settings, choose *Administration* > *Options and Settings*. In the navigation pane of the *Options* dialog box, click *SAP* and then click the *Debugging and Tracing* tab on the right. The following settings are provided:

- Debugging and tracing: Enable for business user
  Usually, you examine actions performed on a screen when you are logged on to the cloud solution with your development user. However, you can also enable debugging and tracing to examine actions that are performed by a specific business user.
  If you enable this option the breakpoints that you set in your script file are only relevant for this specific business user. This means that the debugger only stops at a breakpoint when the script file is executed by this specific user. The debugger does not stop at a breakpoint when the script file is executed by another user.

  > **i Note**
  >
  > Change the setting of this option only when you are in design mode.

- Before starting the debugger
  Before you start debugging, the system prompts you to save changes and to refresh and activate your solution. Here you can change this setting, for example, you can enable the system to save, refresh, and activate automatically every time you start debugging.

> **i Note**
>
> The other debug settings in the *Options* dialog box are default Microsoft Visual Studio settings, which do not have any impact on the studio's debugger.

8.17.2.3 Debugging a Solution

This simple process explains how you can easily debug your solution. For information about the functions that you can use for debugging, see Debugging Quick Guide [page 629].

Prerequisites

- You have created a solution, defined a business object, and implemented the business logic in a script file.
- You have created a screen for your business object.

Process Flow

1. You open the script file in the code editor and set breakpoints.
2. You start debugging.

3. You open your screen, for example, by using the preview function, to observe the runtime behavior of your solution.
   You can also debug your solution when you log on to the SAP cloud solution and select the relevant screen.
4. On the screen, you perform the actions that are implemented in your script file.
   When the debugger hits a breakpoint, it breaks the execution of your solution and the system brings the code editor with the corresponding script file to the front.
   You can now step through the code, display variable values, and analyze the tracing information.
5. When you have found an error, you stop debugging and correct the error.
6. You start debugging again. Make sure that you have reactivated the solution and preview your screen again.

8.18 Tracing

8.18.1 Tracing Quick Guide

You can use tracing to record information about your code as it is being executed. From the *Trace Explorer*, you can run a local trace, run a trace for an end user, use incident traces for troubleshooting purposes, and view the results of a recorded trace.

> **i Note**
>
> If you have started the debugger tracing is turned on by default. You can start tracing separately when the debugger is not started.

You can access the tracing functions as follows:

- To enable the *Trace Explorer* tool window, click *Debug* > *Windows* > *Trace Explorer*.
  The *Trace Explorer* provides a view of active and completed traces.
- To enable the *Debug* toolbar, click *View* > *Toolbars* > *Debug*.
  The *Debug* toolbar provides buttons that allow you to start and stop tracing separately. When you have already started the debugger, these tracing buttons are inactive.

Business and Technical Background

Tracing

You can use tracing to record information about your code as it is being executed, including variable values, informational messages, and error messages. By inspecting the information recorded by a trace, you can obtain a better understanding about how your code works and troubleshoot problems.

For more information about tracing, see here [page 641].

Trace Class and Methods

You can use the methods defined by the `Trace` class in the `ABSL` namespace to record information about the state of your code. Method calls from the `Trace` class are evaluated and recorded when a trace is active in the system. The `Trace` class provides methods to record informational messages (`Trace.Info`) and error messages (`Trace.Error`).

For more information about the `Trace` class and methods, see here [page 643].

Tasks

Run a Local Trace

You can activate tracing on any system that you can log on to with your developer user account. When a trace is active, script files are logged as they are executed, and methods from the `Trace` class are evaluated and recorded.

For more information about this task, see here [page 646].

Run a Trace for a Business User

You can activate tracing for a business user. When tracing has been activated for a business user, the code that is executed by that user is traced. Because tracing is a developer feature, business users cannot enable tracing directly.

For more information about this task, see here [page 647].

Working with Incident Traces

When a user reports an incident in a productive system, you can ask them to reproduce the incident and include trace information. This information can help you track down the source of the issue that the user is experiencing. Incident traces contain the same information as local traces, but are tied to the lifecycle of the incident.

For more information about this process, see here [page 648].

View Trace Results

When a trace has been completed, you can view the recorded information in the *Trace Explorer*.

For more information about this task, see here [page 648].

8.18.2 Tracing

Overview

You can use tracing to record information about your code as it is being executed, including variable values, informational messages, and error messages. By inspecting the information recorded by a trace, you can obtain a better understanding about how your code works and troubleshoot problems.

When you have started tracing, some information is traced automatically, including the identity of each script file that is executed. You can enhance the information that is recorded during a trace by adding methods from the `Trace` class to your script files. This way, you can record informational and error messages, and output variable values.

Prerequisites

You are familiar with creating business objects and working with the scripting language.

Features

Local Traces

Local traces are the traces that are created when you start tracing on the system that you are logged on with your developer user account. You can use local traces to evaluate code that is in development, or to investigate issues that have been reproduced on a local system.

- **Local Traces for Developer Users**
  When you start tracing, the script files that you run under your developer user account are recorded and the methods from the `Trace` class are evaluated and recorded.
  For more information, see Run a Local Trace [page 646].
- **Local Traces for Business Users**
  You can configure tracing to record the actions of a business user instead of tracing the actions performed under your developer user account. This enables you to observe workflows that are performed by a business user in the cloud solution. Business users cannot activate tracing directly because tracing is a developer feature that is accessed only in the studio. When you run a trace for a business user, the script files and the trace statements in the code that are executed by the business user are evaluated and recorded.
  For more information, see Run a Trace for a Business User [page 647].

Incident Traces

When business users encounter errors in a production system, they can use the incident reporting process to report the error and request assistance. Business users can attach trace information to incidents, and you can use this information to help troubleshoot the cause of the incident.

You work with incident traces the same way as you work with local traces. However, incident traces are tied to the life span of an incident. Incident traces are created when the incident is submitted and are deleted when the incident is closed. When you use the trace information that is attached to an incident, it is important to compare it to code from the same version of the system that was used to report the incident. Otherwise, the code paths may have changed and trace statements may be different.

For more information, see Working with Incident Traces [page 648].

Performance

Trace information is only recorded while an active trace is running. When tracing is not active, no system resources are allocated for tracing. The methods from the `Trace` class are only evaluated when a trace is active; therefore you can include them in your script files without incurring any performance penalty during the course of regular operation.

> → Tip
>
> Tracing can record a potentially large amount of information; therefore it is recommended to start tracing for limited periods of time. Otherwise, the amount of data recorded by the trace may become unmanageable. Tracing is automatically stopped if it is left running.

Interpreting Trace Results

In the *Trace Explorer* window, you can view information about the session, sequence, and script files that are called, as well as the results of evaluating methods from the `Trace` class that have been added to script files, and information about other errors that have occurred.

> i Note
>
> Tracing can record several sessions in parallel at a time. A session typically corresponds to the actions performed on one screen. In contrast to this, the debugger can evaluate only one session in parallel at a time.

As each session, sequence, and script file is called, a trace message is recorded in a tree structure in the trace details pane. The trace messages are numbered in the order that they were called, and are nested under the calling parent session, series or script file as follows:

- **Session**
  Each session is displayed in order.
- **Sequence**
  Each sequence is displayed in order.

- **Script file**
  Each script file is displayed in the order it was called, and may be nested if it was called from another script file. This view displays a summary of all of the trace messages for this script file. If any of the trace messages for the script file are error messages, then an error icon is displayed. This enables you to quickly identify which script files contain error messages without having to manually view the details about each script file.

You can view details about the trace messages recorded for each script file by double-clicking the script file entry in the *Trace Explorer* window. This displays the script file in the *Trace Details* pane. The *Trace Details* pane contains a list of the trace messages that were recorded for the script file. The first item in this list is a message that identifies the script file. The rest of the list contains the results of calling methods from the `Trace` class, and errors that may have occurred. The details for the trace messages are as follows:

- **Status**
  This defines whether the message is an error or for information purposes.
- **Type**
  This defines the type of message. For example, every script file is identified by a message whose type is `"this" object ID`. Messages that were generated by method calls from the `Trace` class are identified by a message of type `user-defined trace command`.
- **Number**
  The order that this trace message was processed for this script file.
- **Name**
  The name of the message. For messages recorded by the `Trace` class, this displays the first parameter passed to the `Trace.Info` and `Trace.Error` method calls.
- **Value**
  The value of this message. For object identification messages, this can be the alternative key or the UUID of the object. For `Trace` class messages, this is the value of the second parameter passed to `Trace.Info` and `Trace.Error` method calls.

Related Information

Trace Class and Methods [page 643]
Tracing Quick Guide [page 639]

8.18.3 Trace Class and Methods

Overview

You can use the methods defined by the `Trace` class in the `ABSL` namespace to record information about the state of your code. Method calls from the `Trace` class are evaluated and recorded when a trace is active in the system. The `Trace` class provides methods to record informational messages (`Trace.Info`) and error messages (`Trace.Error`).

Trace Class

The `Trace` class provides methods that you can use to record information about the state of your code.

Importing the Trace Class

You import the `ABSL` namespace to access the `Trace` class.

Example

```
import ABSL;
```

Trace.Error Method

You use the `Error` method to record information about errors. There are two overloads of the `Error` method, taking one or two `String` parameters. The results of evaluating these methods are displayed in the *Trace Details* pane and are marked by an error icon.

Syntax	`Trace.Error(String);` `Trace.Error(String, String);`
Example	`Trace.Error("Creation failed");` `Trace.Error("Creation failed", this.Identifier.Content);`

Trace.Info Method

You use the `Info` method to record information that is not an error case. There are two overloads of the `Info` method, taking one or two `String` parameters. The results of evaluating these methods are displayed in the *Trace Details* pane and are marked by an information icon.

Syntax	`Trace.Info(String);` `Trace.Info(String, String);`
Example	`Trace.Info("Test triggered");` `Trace.Info("Test triggered", this.Identifier.Content);`

Example

```
import ABSL;
Trace.Info("Start Test", this.Identifier.Content);
Trace.Error("Invalid parameter", this.Identifier.Content);
Trace.Info("End Test");
```

Trace.Runtime Method

You use the `Runtime` method to record the lapsed time since the very first call of this method or the latest reset call. There are two overloads of the `Runtime` method, taking one `String` parameter or a `String` parameter and an `Indicator` parameter. If the `Indicator` is set to `true`, the runtime is reset and the measurement will show zero again.

The results of evaluating these methods are displayed in the *Trace Details* pane and are marked by an runtime icon. The value column will show the runtime used up in seconds.

Syntax	`Trace.Runtime(String);` `Trace.Runtime(String, Indicator);`
Example	`Trace.Runtime("Runtime measurement started", true);` `Trace.Runtime("Runtime used up to now");`

Example

```
import ABSL;
Trace.Runtime("Start Measurement", true);
```

```
Trace.Runtime("Current time used up");
```

Related Information

Tracing [page 641]
Tracing Quick Guide [page 639]

8.18.4 Tasks

8.18.4.1 Run a Local Trace

Overview

You can activate tracing on any system that you can log on to with your developer user account. When a trace is active, script files are logged as they are executed, and methods from the `Trace` class are evaluated and recorded.

Prerequisites

You must know how to run your script files, and you may want to add methods from the `Trace` class to your script files to record custom information.

Procedure

1. To begin recording trace information, in the *Trace Explorer* window, click the *Start Tracing* button.
2. Run the script files that you want to trace.
3. To stop recording trace information, in the *Trace Explorer* window, click the *Stop Tracing* button.
4. To view traces, in the *Trace Explorer* window, click the *Local Traces* tab.
5. To update the list of traces, click the *Refresh Traces* button.
   The completed trace appears in this list, identified by the time that trace was started, and the ID of the user executing the code.

Related Information

Run a Trace for a Business User [page 647]

Trace Class and Methods [page 643]

8.18.4.2 Run a Trace for a Business User

Overview

You may want to trace a workflow that can be run by a business user but cannot be run by a developer user. For example, you must be logged on to the SAP cloud solution as a business user to access some workflows. Since business users cannot activate and deactivate tracing, developer users can run a trace on their behalf through the *Trace Explorer*. When a developer user activates a trace for a business user, the code that is executed by the business user is traced.

Prerequisites

To record custom information, you have to add methods from the `Trace` class to your script files.

Procedure

1. To open the trace settings dialog, click *Administration* > *Options and Settings*. In the navigation pane of the *Options* dialog box, click *SAP* and then click the *Debugging and Tracing* tab on the right.
2. Select *Enable for business user*.
3. In the text box, enter the ID of the business user to enable tracing, and click *OK*.
   For example, enter **`PGREENE`** to enable tracing for the business user Peter Greene.
4. In the *Trace Explorer* window, click the *Start Tracing* button.
   A dialog is displayed that confirms that the trace has been activated for the business user.
5. Log on to the system as the business user and run the workflow you want to trace.
6. In the *Trace Explorer* window, click the *Stop Tracing* button.
   A dialog is displayed that confirms that the trace has been deactivated for the business user.
7. In the *Trace Explorer* window, click the *Local Traces* tab.
8. To update the list of traces, click the *Refresh Traces* button. The trace appears with the time stamp of when the trace was started and the ID of the business user.
   To view a trace, see View Trace Results [page 648].
   To delete a trace, select one or more from the list and click *Delete Selected Traces*.

Related Information

Run a Local Trace [page 646]
Trace Class and Methods [page 643]

8.18.4.3 Working with Incident Traces

Overview

When a user reports an incident in a productive system, you can ask them to reproduce the incident and include trace information. This information can help you track down the source of the issue that the user is experiencing. Incident traces contain the same information as local traces, but are tied to the lifecycle of the incident.

Prerequisites

You use the same version of the system that was used to report the incident.

Process Flow

1. A user encounters an error in the cloud solution and reports an incident to an administrator.
2. The administrator returns the incident and requests trace information.
3. In the *Incident* editor, the user clicks *Start Trace* and repeats the steps that caused the incident.
4. In the studio, the incident trace appears in the *Incident Trace* tab of the *Trace Explorer*.
5. Developer users with the relevant authorization can view the incident trace in the *Trace Explorer*.
6. The incident trace remains accessible until the incident is closed.

8.18.4.4 View Trace Results

Overview

When a trace has been completed, you can view the recorded information in the *Trace Explorer*.

Prerequisites

A completed local trace or open incident trace exists.

Procedure

1. In the *Trace Selection* area of the *Trace Explorer*, select the type of trace to view:
   - To view local traces, click the *Local Traces* tab.
   - To view traces for open incidents, click the *Incident Traces* tab
2. To display the current list of completed traces, click the *Refresh* button.
3. Select the check box next to the traces that you want to view, and click the *Get Trace Details* button.
   Each selected trace appears in its own tab.
4. Expand the trace details to show the sessions, sequences, and script files that have been recorded for the trace.
   If the script file contains any error messages, it is marked by an error icon. This enables you to easily identify the script files that contain errors. If the script file contains no errors, it is marked by an information icon.
5. Double-click a script file trace message to see detailed trace information for that script file in the *Trace Details* pane.

Related Information

Tracing [page 641]
Run a Local Trace [page 646]
Run a Trace for a Business User [page 647]
Working with Incident Traces [page 648]

8.19 Dump Analysis

8.19.1 Dump Analysis Quick Guide

You can use the *Dump Analysis* tool window to list the runtime errors that have occurred in the tenant to which you have logged on. The tool window presents the dump information in a reader friendly format and supports you in finding the root cause of a dump.

If the *Dump Analysis* tool window is not visible in the studio, you can access it by clicking *View* > *Dump Analysis*.

Business background

What is a dump?

If a severe error occurs during execution of a solution, the runtime cannot continue to execute the solution and the solution is terminated. In such a case a dump is written, which is a data record about the solution's state

when it had been terminated. Severe runtime errors occur, for example, when an unassigned reference or a division by zero is used or the generated code contains syntax errors. Typically, the runtime error messages that are displayed on the screen do not contain detailed information about the root cause of the dump. Runtime errors that occur in background programs or asynchronous processes are even harder to detect, since there is no immediate feedback to the user.

Tasks

Check for Dump Entries

In the *Dump Analysis* tool window, select one of the following options:

- *Today*
  All dumps that occurred today are listed.
- *Yesterday*
  All dumps that occurred yesterday are listed.
- *All*
  All dumps that occurred are listed. The period covered depends on how long dumps are kept in the system. The dump entries in the list are sorted by date.
- *Advanced*
  In the corresponding dialog box, you can specify a date, a date range, a user name or a solution name. The tool window will then list all dumps that meet these criteria.

> → Tip
>
> The dump list is not updated automatically. To update the list, click *Refresh* or any of the other buttons in the tool window. If you click *Reset*, the filter is cleared and the list shows the dump entries of the current day.

> i Note
>
> Dumps caused by the following runtime errors are **not** listed in the *Dump Analysis* tool window:
>
> - The business object element has been deleted or renamed but the element is still used on the user interface (UI).
> - The data string in the business object element is too long for the database.

Filter Dump Entries

If the list contains a large number of dump entries, you can set a filter to reduce the number of entries by specifying a date, a date range, a user name or a solution name.

1. In the *Dump Analysis* tool window, click *Advanced*.
   The *Advanced Selection* dialog window is displayed.
2. Enter the relevant data and click *Search*.
   Only the dump entries are listed in the *Dump Analysis* tool window that meet the filter criteria.

3. Optional: To clear the filter, click *Reset* in the *Dump Analysis* tool window.

Evaluate Dump Entries

1. In the *Dump Analysis* tool window, select a dump entry.
   You can check the following:
   - Time when the error occurred
   - User who executed the solution when the error occurred
   - Error type
   - Solution name
   - Path to the script file that caused the error
   - Line and column of the script file where the error occurred
     If these fields contain numbers in black, the position of the error in the script file has been identified and you can navigate to this position in the script file. However, it is not always possible to show the correct position of the error. For example, if the script file has been changed, cleaned, or deleted after the error has occurred, the information about the previous position is displayed in grey. If the position cannot be determined at all, the fields contain the information `n/a`.
2. Double-click the dump entry to open the script file in the editor.
   The cursor is positioned in the line of the script file where the error occurred.
   If the script file where the error occurred has been called by another script file, you can see this information in the *Call Stack* area of the *Dump Analysis* tool window, .
3. Optional: In the *Call Stack* area, click the relevant line to open the script file that has called the other one.
   The cursor is positioned in the line of the script file where the error occurred.

> **i Note**
>
> You can navigate from a line in the tool window to the corresponding script file, if the following conditions are met:
>
> - The current user is assigned to the customer ID for which the solution has been created..
> - The solution or script file has not been deleted.

8.20 Translation

8.20.1 Translation Quick Guide

The studio provides functions that enable the translation of business configuration content texts and on-screen texts into one of the languages supported by the cloud solution. The translation functions enable you to adapt a solution created using the studio for a specific country or region. You can export the text of specific project items to a file in XML Localization Interchange File Format (XLIFF). After the XLIFF file is translated, you can then import the translated file into the studio and check the result. You can also monitor the translation status of project items.

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

The following table gives an overview of the project items from which you can extract text for translation and the specific text elements that can be translated.

Texts That Can Be Extracted for Translation

Project Item	Text Element
User interface (UI) component (`*.uicomponent`) based on a custom business object (`*.bo`)	• Title of floorplan and group • Column and menu header • Labels for field, checkbox, and radio button • Definition, description, instruction, and information • Other on-screen texts such as button text, link text, list item text, explanation text, general text, tab strip text, quick info text, tree node text, bullet list item text, technical text The system does not support the extraction of message texts to an XLIFF file.
Business object (`*.bo`)	• Label of a business object field • Tooltip of a business object field The system does not support the extraction of message texts to an XLIFF file.
Business object extension (`*.xbo`)	• Label of an extension field • Tooltip of an extension field The system does not support the extraction of message texts to an XLIFF file.
Business adaptation catalog (BAC) element (business option, business topic)	• Description of a business topic • Description of a business option • Scoping question/statement of a business option • Details: *Overview* and *Relevance* texts
Business configuration (BC) set (`*.bcc`)	Description of a field value
Code list data type (*.codelist)	Description of code list value
Data mashup, HTML mashup, URL mashup (`*.MC.uimashup`)	On-screen texts

Business and Technical Background

XML Localization Interchange File Format (XLIFF)

XML Localization Interchange File Format (XLIFF) is a standard XML-based file format for localization, specifically designed for exchanging bilingual content between systems. This format enables translators to concentrate on the text to be translated by extracting translatable text and separating it from non-translatable text. XLIFF is beneficial as a single format which can be used for translating different kinds of files. With all the advantages of XML-based processing, this format aims to standardize localization and is supported by a wide variety of translators.

For more information, see XML Localization Interchange File Format (XLIFF) [page 654].

Managing Translation

When you create a solution in the studio, you can translate business configuration content and other solution content into another language. To translate business configuration content and on-screen texts, you must export the translatable content to an XLIFF file on your local computer. Once the XLIFF file is translated, you can import the translated file into the studio and check the translation status and the result.

For more information, see Managing Translation [page 657].

Tasks

Export Text for Translation

In the studio, you can extract translatable texts from specific project items in a solution. You export this text to a file in XML Localization Interchange File Format (XLIFF) to your local computer for translation into one of the languages supported by the SAP cloud solution. You specify the target language when you export the text.

For more information, see here [page 658].

Import Translated Text

After you have exported text of the translatable project items in a solution to a file in XML Localization Interchange File Format (XLIFF) and it has been translated into the target language, you can import the translated XLIFF file from your local computer to the solution and test the results of the translation in the target language in the SAP cloud solution.

For more information, see here [page 660].

Check Translation Status

You can check the translation status of a project, business configuration content and a UI component by viewing the total number of translatable text strings and translated text strings.

For more information, see here [page 661].

Translate Message Texts

1. Translate message texts that you defined for a business object or a business object extension in the corresponding `.bo` or `.xbo` file by using the following syntax based on a sample message:

```
message NoValidSite    text "Enter a valid site";
message NoValidSite_DE text "Geben Sie einen gültigen Ort ein";
message NoValidSite_ZH text "输入有效数值";
```

2. In your action or event script file, raise the language-dependent message using the `var` expression by using the following syntax:

```
var userLanguageCode;
...
userLanguage = Context.GetCurrentUserLanguage().ToString();
if ( userLanguage == "DE" ) { raise NoValidSite_DE.Create("E"); }
else if ( userLanguage == "ZH" ) { raise NoValidSite_ZH.Create("E"); }
else { raise NoValidSite.Create("E"); }
```

8.20.2 Business and Technical Background

8.20.2.1 XML Localization Interchange File Format (XLIFF)

Overview

XML Localization Interchange File Format (XLIFF) is a standard XML-based file format for localization, specifically designed for exchanging bilingual content between systems. This format enables translators to concentrate on the text to be translated by extracting translatable text and separating it from non-translatable text. XLIFF is beneficial as a single format which can be used for translating different kinds of files. With all the advantages of XML-based processing, this format aims to standardize localization and is supported by a wide variety of translators.

XLIFF File

As XLIFF is an XML-based file format, it begins with an XML declaration. After the XML declaration comes the XLIFF document itself, enclosed within the `<xliff>` element. An XLIFF document is composed of one or more

sections, each enclosed within a <file> element. Each <file> element corresponds to a file which has to be translated and contains information about the source language and the target language. It could be a project, a business configuration, a project item, such as a floorplan or a business configuration element, such as a BC set or a BAC element. The file element consists of a <header> element and a <body> element.

The following is an example of the basic structure of an XLIFF file:

Example

```
<?xml version='1.0' encoding='utf-8'?>
<xliff xmlns="urn:oasis:names:tc:xliff:document:1.2" version="1.2">
  <file datatype="plaintext" date="2011-12-02T06:22:58Z" original="self"
source-language="en" target-language="de">
    <header>
    </header>
    <body>
      <group>
        <trans-unit>
        </trans-unit>
      </group>
    </body>
  </file>
</xliff>
```

The <header> element contains metadata about the file which has to be translated. This information depends on the level at which you want to translate.

The following is an example of a <header> element in an XLIFF file exported at UI component level:

Example

```
<header>
  <phase-group>
    <phase company-name="SAP AG" date="2011-12-02T06:22:58Z" phase-name="file-
creation" process-name="file-creation" tool-id="urn:x-sap:byd:xrep:xliff"/>
  </phase-group>
  <note from="SAP">In order to provide additional context/meta data for
translators, The strings in this file are
grouped according to UI</note>
  <tool tool-company="SAP AG" tool-id="urn:x-sap:byd:xrep:xliff" tool-
name="ByD XRep XLIFF Interface" tool-version="1.1.0"/>
  <sxmd:metadata xmlns="urn:x-sap:mlt:tsmetadata:1.0" xmlns:sxmd="urn:x-
sap:mlt:xliff12:metadata:1.0">
    <object-name>BY003505</object-name>
    <collection>/BY003505/BC</collection>
    <domain>BAC</domain>
    <developer>TESTUSER</developer>
    <description>BY003505 Translation XLIFF File</description>
    <origin>/BC_BY003505/SRC/BusinessConfiguration.xml</origin>
  </sxmd:metadata>
</header>
```

The <body> element contains the extracted translatable data from the file in the form of translation units represented by <trans-unit> elements. The translatable data within <trans-unit> elements is organized into source and target paired elements. A <source> element stores the source text, and a <target> element stores the translated text. The <target> elements are not mandatory. These <trans-unit> elements can be grouped recursively in <group> elements. For example, If you translate at the project level, the <header> element contains information about the project and each <group> element contains information about an individual UI component.

The following is an example of a <trans-unit> element translated from English to German:

Example

```
<trans-unit  xmlns:sap="urn:x-sap:sls-mlt"  datatype="plaintext"  id="UICT-/
BY003505/USINESSOBJECT1_QA.QA-637f86765fbf498ba2cc6017dd6696f0"
maxwidth="255"  resname="637f86765fbf498ba2cc6017dd6696f0"  restype="x-Text"
sap:sc="XTXT ">
  <source>Save</source>
  <target >Speichern</target>
</trans-unit>
```

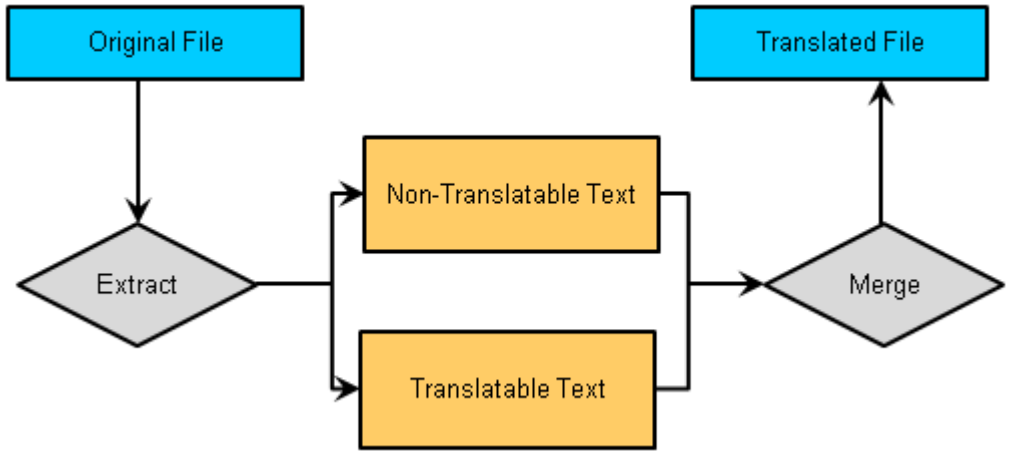
Note

XLIFF is bilingual as each translation unit offers one <source> element and one <target> element. This makes the overall model simple and easy to handle.

Extraction and Merge Principle

Translation in XLIFF is based on the concept of extracting the translatable text from the original file, and merging it back in place after translation has been done.

The extraction and merge principle is illustrated in the following diagram:



XLIFF Extraction and Merge Principle

All translatable texts and non-translatable texts are extracted from the original file. The parts that are non-translatable are preserved in a temporary file. The translatable parts are stored separately and translated. After translating the translatable texts, both the parts are merged, resulting in the translated file.

Related Information

Translation Quick Guide [page 651]

8.20.2.2 Managing Translation

Overview

When you create a solution in the studio, you can translate business configuration content and other solution content into another language. To translate business configuration content and on-screen texts, you must export the translatable content to an XLIFF file on your local computer. Once the XLIFF file is translated, you can import the translated file into the studio and check the translation status and the result.

Prerequisites

- You have opened a solution in the studio that contains content that needs to be translated.
- To test translated solution content, you must be assigned the Business User role and have access rights for the work center views where the content is displayed. For more information, see User Setup Quick Guide [page 33].

Process Flow

1. You extract the translatable text strings of all translatable project items in a solution or a specific project item and export the text to an XLIFF file that you save on your local computer.
   For more information, see Export Text for Translation [page 658].
2. The XLIFF file is translated into the target language using a translation tool that supports XLIFF or an XML editor.
   For more information about XLIFF, see XML Localization Interchange File Format (XLIFF) [page 654].
3. You import the translated XLIFF file into the studio.
   For more information, see Import Translated Text [page 660].
4. You activate the translated project items.
   If you have translated `.uicomponent` files, you need to update the metadata in the UI designer. To do this, proceed as follows:
   1. Right-click the translated `.uicomponent` file and select *Open in UI Designer*.
   2. Select *Tools* > *Update Metadata*.
   3. Click the *Save and Activate* button ( ).
5. Optional: You check the translation status of the solution content.
   For more information, see Check Translation Status [page 661].
6. Optional: To test translated business configuration content, you deploy the business configuration to ensure that you can view the most recently translated business configuration set values in the SAP cloud solution.
   For information about business configuration content, see Business Configuration Quick Guide [page 362].
7. You log on to the SAP cloud solution as a business user and make sure that you select the target language into which the solution content was translated. You test the translated text by viewing the relevant screens and making sure that all texts appear and are displayed correctly.

8. You request a test deployment. For more information, see Administration Quick Guide [page 159].

Related Information

Translation Quick Guide [page 651]

8.20.3 Tasks

8.20.3.1 Export Text for Translation

Overview

In the studio, you can extract translatable texts from specific project items in a solution. You export this text to a file in XML Localization Interchange File Format (XLIFF) to your local computer for translation into one of the languages supported by the SAP cloud solution. You specify the target language when you export the text.

Prerequisites

You have opened a solution in the studio that contains content that needs to be translated.

Procedure

1. Extract the translatable text strings of all translatable project items in a solution or a specific project item by doing the following:
   1. In the *Solution Explorer*, right-click one of the following items:
      - Project node (to extract the translatable text strings of all project items that support translation)
      - Business object (`*.bo`)
      - Floorplan (`*.uicomponent`)
      - Business adaptation catalog element (`*.bac`)
      - Business configuration set (`*.bcc`)
      - Business object extension (`*.xbo`)
   2. Select *Translation* > *Export Text*.
      The *Export Text for Translation* dialog box opens.
2. In the *Export Text for Translation* dialog box, select the target language to which you want to translate the text.
3. The *Add Standard Translation* option is preselected. If you do not deselect it, the system inserts standard translations of common UI texts into the XLIFF file when you export your texts for translation.

4. Enter a name for the XLIFF file.
   You can use the default file name or provide a new name. The default file name is in the format `Technical Name_Target Language Code.xlf` where `TechnicalName` is the name of the project or project item that you selected. `Target Language Code` is an abbreviation of the target language you selected. This code will be added to the generated XLIFF file.

   Examples of Default File Names for Exported XLIFF Files

Item	Technical Name	Target Language	XLIFF File Name
Project	`Y3S15QDEY_`	German	`Y3S15QDEY_DE.xlf`
Object Instance Floorplan	`BusinessObject1_OIF.OIF.uicomponent`	Italian	`BusinessObject1_OIF_IT.xlf`
Object Work List	`BusinessObject1_OWL.OWL.uicomponent`	Portuguese	`BusinessObject1_OWL_PT.xlf`
Work Center View	`BusinessObject1_WCView.WCView.uicomponent`	Spanish	`BusinessObject1_WCView_ES.xlf`
Business Adaptation Catalog Element	`BACElements1.bac`	Chinese	`BACElements1_ZH.xlf`
Business Configuration Set	`CODE.bcc`	French	`CODE.bcc_FR.xlf`

5. Select the location on your computer where you want to save the exported XLIFF file.
   You can save the file in the default location or specify a new location.
6. Click *OK*.
   The file is saved in the specified folder on your local computer.

Result

You have exported the solution content to an XLIFF file on your local computer. You can now translate the XLIFF file into the target language using a translation tool that supports XLIFF or an XML editor. For more information about XLIFF, see XML Localization Interchange File Format (XLIFF) [page 654]. You can also open the XLIFF file in the studio: from the *File* menu, select *Open* > *File* and navigate to the location of the XLIFF file you exported. The file opens in a document window in the code editor.

Related Information

Translation Quick Guide [page 651]

8.20.3.2 Import Translated Text

Overview

After you have exported text of the translatable project items in a solution to a file in XML Localization Interchange File Format (XLIFF) and it has been translated into the target language, you can import the translated XLIFF file from your local computer to the solution and test the results of the translation in the target language in the SAP cloud solution.

Prerequisites

- You have exported an XLIFF file and saved it on your local computer.
  For more information, see Export Text for Translation [page 658].
- You have translated the XLIFF file into the target language.
  For more information about XLIFF, see XML Localization Interchange File Format [page 654].

Procedure

1. In the *Solution Explorer*, right-click the project, floorplan, business adaptation catalog element, or business configuration set for which you want to import translated text and select *Translation* > *Import Text*.
   The *Import Translation* dialog box opens.
2. Select the location on your computer where you have saved the translated XLIFF file.
3. Click *OK*.
4. In the *Solution Explorer*, activate the translated project items by right-clicking the project or the project items and selecting *Activate*.
5. Optional: If you have imported translated business configuration content, deploy the business configuration by right-clicking the project node and selecting *Deploy Business Configuration*.
   You must deploy business configuration content to ensure that you can view the most recently translated business configuration set values.
6. Log on to the SAP cloud solution as a business user and make sure that you select the target language into which the solution content was translated.
7. Test the translated text in the SAP cloud solution by viewing the relevant screens and checking that all texts appear and are displayed correctly.

Related Information

Translation Quick Guide [page 651]

8.20.3.3 Check Translation Status

Overview

You can check the translation status of a project, business configuration content and a UI component by viewing the total number of translatable text strings and translated text strings.

Prerequisites

You have opened the solution in the studio for which you want to check the translation status.

Procedure

1. In the *Solution Explorer*, right-click the project, floorplan, business configuration set, or other project item for which you want to check the translation status and select *Translation* > *Check Status*.
   The *Check Translation Status* dialog box opens.
2. Review the total number of translatable text strings and translated text strings, as well as the language to which the text strings have been translated.
   If you check the translation status at project level, the *Check Translation Status* dialog box displays the translation status of all project items in the solution for which translation is supported.
3. Do one of the following:
   - To export any texts that still need to be translated into one or more languages, select a language and click *Export to XLIFF*.
     The *Export Text for Translation* dialog box opens. For more information, see Export Text for Translation [page 658].
   - If all texts have been translated, click *OK*.
     The *Check Translation Status* dialog box closes.

> **i Note**
>
> If you check the translation status at project level, you can only export text at the project level and not for individual project items.

Related Information

Translation Quick Guide [page 651]

9 User Interface Designer

9.1 User Interface Designer Overview

The user interface designer (UI designer) is integrated into the studio. You use it to design the user interface of your solution by modifying the screens that you have created in the studio, for example, quick activity floorplan, object work list, or work center view. You can also change or enhance SAP floorplans. Further, you can also use the UI designer to design screens for mobile devices.

The UI designer contains a number of different tools and functions that you can access from the menu or on tabs at the bottom of the UI designer screen. For more information, see Features of the User Interface Designer [page 664].

Availability

> ⚠ Caution
>
> This documentation covers the SAP Cloud Applications Studio (“studio”) as a whole; therefore it describes **all** features that are available within the studio. Not all of these features are available for every cloud solution from SAP. Before using a particular feature, please consult the Feature Matrix to determine whether the feature is available for the cloud solution from SAP for which you are creating your solution.
>
> For details, see Feature Matrix [page 27].

Prerequisite

You have created a solution and created one or more screens in the studio. For more information, see Generate Screens for a Business Object [page 429].

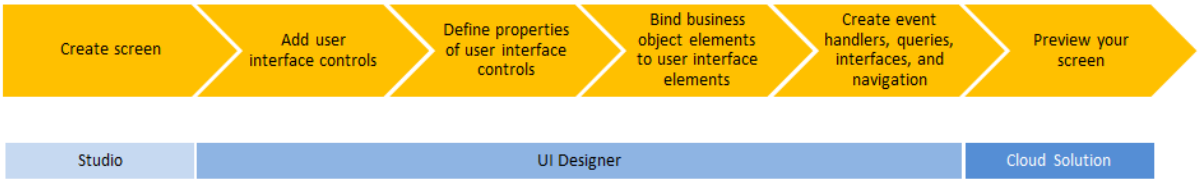
What you can do in the UI designer

1. **Model custom floorplans in the UI designer**
   You can modify the screens that you have created in the studio by adding panes and controls as well as creating event handlers, queries, interfaces, and navigation. You can bind the controls you have added to business object elements so that when a particular control is used, data is retrieved from the business object.
   For more information, see Floorplan Modification Quick Guide [page 719].

2. **Modify SAP floorplans**
   You can modify SAP floorplans, for example, add a view to a work center or an embedded component to a floorplan. For information on which changes you are allowed to make, see Change Transactions Quick Guide [page 755].
   You need to use anchors to make sure that the changes are included in every upgrade of the SAP cloud solution. You can use existing anchors or create a new anchor. For more information, see Anchors [page 759].
3. **Adapt floorplans for mobile devices**
   You can also modify floorplans for use on a mobile device. For more information, see .

Process Flow

The following graphic shows how and where you create and design the screens of your solution:



How you design screens for a solution

To create a screen, do the following:

1. Create screens such as an object work list, a fact sheet, or quick activity floorplan in the studio.
   For more information, see Generate Screens for a Business Object [page 429].
2. In the UI designer, drag the required panes and controls to design the layout of the screen.
   For more information, see .
3. Define the properties of each user interface control.
   For more information, see the **Property Explorer** section in Features of the User Interface Designer [page 664].
4. Bind the business object elements with the user interface elements.
   For more information, see Bind Data Elements to Business Object Elements [page 731].
5. Define the behavior of your solution by creating the following:
   - Event handlers. For more information, see Create an Event Handler [page 747].
   - Navigation. For more information, see Configure an Object-Based Navigation in a Source Floorplan [page 744] and Configure an Object-Based Navigation in a Target Floorplan [page 742].
   - Queries. For more information, see Configure a Query [page 741].
   - Scripts. For more information, see Write a Script [page 746].
6. Preview your screens to check your changes. For more information, see .

   > **i Note**
   >
   > Before activating your screens, we recommended that you run the checkmate tool using the *Run Checks* button on the top tool panel on your screen.

7. Save and activate your screens.

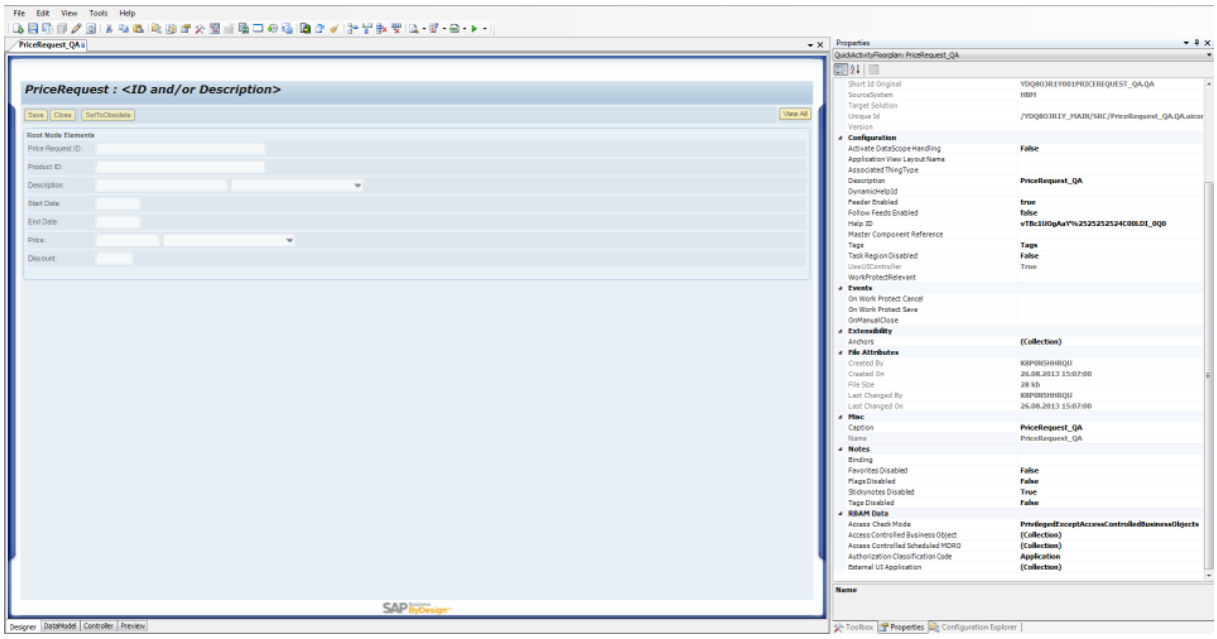
9.2 Getting Started with the User Interface Designer

9.2.1 Features of the User Interface Designer

Overview

This document provides a brief overview of the common features available in the user interface designer (UI designer) to design the user interface.

This is what the UI designer looks like:



User Interface Designer

File Menu

Menu Entry	Description
Close	Closes a user interface component.
Import	Allows you to import user interface component from a file in the local disk. File can be of type `.User Interfacecomponent` file or a `.xml` file.
Save	Saves a user interface component in the repository.
Save All	Saves all the user interface components that are being edited.

Activate	Activates a user interface component, so that the model is available to all users who are accessing the repository.
Exit	Exits the UI designer.

View Menu

The *View* menu allows you to display or close the following tool windows:

- **Configuration Explorer**
  The *Configuration Explorer* displays the repository content in the UI designer. You can filter the content by a selected component type.
  The last level in the hierarchy is the user interface component. You can double-click to open the component. The tool tip for the component shows the component type and the complete path which acts as a unique identifier.
- **BO Browser/Data Model**
  The *BO Browser* appears in different places in the UI designer:
  - As part of the BOBrowser / DataModel tool
  - In the *Data Model* tab of a component
  - In the *Controller* tab of a component

  In the *BO Browser* tab, the assigned business object (BO) or enhanced controller object (ECO) is displayed as a tree structure with all available nodes and fields. You can:
  - Choose to display only the elements, queries, actions or everything
  - Add elements and attributes, create associations, queries, actions and their parameters
  - Reset the client side cached metadata from the backend (in case the ECO/BO has been updated)
  - Filter for specific categories (if allowed)
  - Search for opened nodes in the hierarchy

  In the *Data Model* tab you can:
  - Create the structure of the data model
  - Create, delete and rename nodes in the data model structure
  - Bind the data model element to the business object element
  - Display the corresponding properties of the nodes in the *Properties Explorer*

  In the *DataModel* tab the data structure is displayed together with various data model elements. The binding status of each of these elements is represented using specific colors. You can also determine the binding status for each element in the *Properties Explorer*.

Color	Binding Status
Blue	Indicates an internal data model element. These are used internally for different purposes. *Binding Status* in the *Properties Explorer* has been set to *Internal*.

Yellow	Indicates an unbound data model element. Data binding has not yet been specified for this element. *Binding Status* in the *Properties Explorer* has been set to *ToBind*.
Green	Indicates a bound data model element. Valid data binding has been specified for this element, or *Binding Status* in the *Properties Explorer* has been set to *OK*.
Red	Indicates a data model element for which the data binding has to be corrected. *Binding Status* in the *Properties Explorer* has been set to *ToCorrect*.

- **Property Explorer**
  It allows you to view and change properties for each element in the user interface model.
  As soon as you select an element in the *Designer* tab, the *DataModel* tab or the *Controller* tab, the corresponding properties of the selected element are displayed in the *Property Explorer*.
  You can edit a few properties, and some are read-only.
  Note the function of the dropdown list above the *Properties* window. You use this list to switch between different components of the floorplan, depending on the item currently selected on the *Designer* tab.
- **Toolbox**
  The *Toolbox* provides the panes and controls that can be used to design the layout of a user interface component in the *Designer* tab.
  The controls and panes available depend on the floorplan. You can drag the panes and controls from the toolbox to the *Designer* tab.
  For more information, see .
- **Document Outline**
  The *Document Outline* provides an overview of all the controls used in the user interface of the selected component.
- **Project Context**
  The *Project Context* provides an overview of the users involved in the creation and use of a particular floorplan.
- **Run Checks**
  Run Checks provides the list of errors in the floorplan. For more information, see Working with the Checkmate Tool.
  For more information, see Working with the Checkmate Tool [page 668]
- **Where Used List**
  The *Where Used List* gives an idea about where exactly a particular user interface component is being used.
- **Output Window**
  The *Output Window* gives an overview of all the changes made to the floorplan that is currently open.
- **Version History**
  The *Version History* gives an idea about the name and type of solution the user is working in, the date and time when the solution was last modified and the user who is responsible for the modifications.
- **OBN Explorer**
  The *OBN Explorer* will provide information about the operations, target floorplans and the inports configured in a particular BO.

You can drag and drop the required OBN onto a user interface control in a source floorplan. The event handler and outport are automatically created; the outport will however need to be configured.

- **Extensibility Explorer**
  The *Extensibility Explorer* is used to modify floorplans that come from layer lower than the layer the user is working on. It displays the anchors assigned for a particular area, such as the toolbar, section group or pane container and the change transactions associated with these anchors.
  All the change transactions that have been made can be viewed in the *Extensibility Explorer*. Change transactions once created, are displayed in the *Unsaved Change Transaction* section, once they are saved they are displayed in the *Saved Change Transaction* section.
  After activation, the change transaction are displayed in the *Published Change Transaction* section.

> → Tip
>
> These tool windows can be moved around, docked, or hidden.

Tools Menu

Menu Entry	Description
Update Metadata	Updates the backend meta data information in the active component.
Options...	Allows the user to personalize the debug and warning message settings.
Cleanup UI Component	Allows the user to delete any unused data structures created by default along with Action Parameters.
Add to Favorites	Adds the floorplan to a favorite list.

Content Area

The content area has the following tabs:

- *Designer*: To design the user interface of an application with the available panes and controls from the *Toolbox*.
- *Data Model*: To bind the user interface elements to a business object element.
- *Controller*: To define the interaction between user interface components by creating event handlers, interfaces, navigation, and queries.
- *Preview*: To preview the changes made to an user interface component, for example, a floorplan.

9.2.2 Working with the Checkmate Tool

Checkmate tool embedded in the UI Designer acts as a guide to help developers model their UI components correctly and consistently. This document lists all the known issues and inconsistencies that you may face while modeling a UI component, and the help document solutions to them.

Enable the Checkmate Tool in the UI Designer

To enable the tool, on the tool panel, select the *Enable CheckMate* checkbox. When you enable this checkbox, the *Run Checks* button gets enabled automatically.

Run the Checkmate Tool in the UI Designer

To run all the consistency and UX checks on the open UI component, click the *Run Checks* button. The check results are displayed at the bottom of the screen in a table. The results show the number of UX and consistency rules present in the UI component.

The check results table consists of the following fields:

- Priority - Displays the priority of the issue. The indicators are as follows:
  - Priority 1 indicates that the issue must be resolved to avoid inconsistencies and instabilities. Such issues are highlighted in red color.

    > ⚠ Caution
    >
    > Do not ignore Priority 1 checkmate errors; resolve them at the earliest.

  - Priority 2 indicates that the issue can affect the stability of the UI component.
  - Priorities 3 and 4 indicate warnings.
- Type - Displays the type of the issue.
- Message/Details - A hyperlink that takes you to the area in the UI component where the issue has occurred.
- Help Document - An external link that explains the issue and provides a solution.
- Status - Indicates whether it is an issue or an exception.
- Exception Reason - Displays the reason for marking the rule as an exception.
- Action - Allows you to mark the rule as an exception.

If a UI component in active status, with the checkmate tool enabled, encounters a Priority 1 issue, it becomes inactive when you try to access, open, or edit it. If you refresh the solution, you can see a red sign against the component in the solution explorer indicating the UI component becoming inactive. Solve the issue and activate the component again. If you decide to activate the component without resolving the issue, make sure that you disable the checkmate tool in the component until the issue is resolved. However, if you access the UI component again, the checkmate tool gets enabled and it may turn the UI component inactive again. Ensure that you activate all the UI components before any assembly or deployment process.

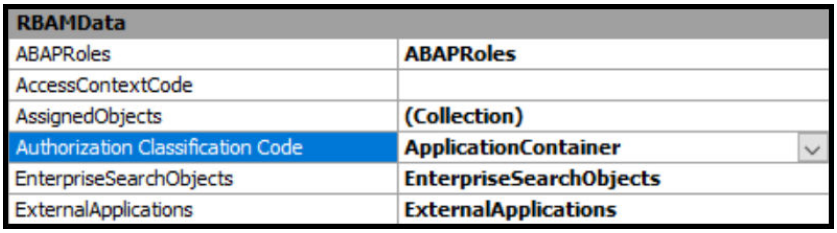
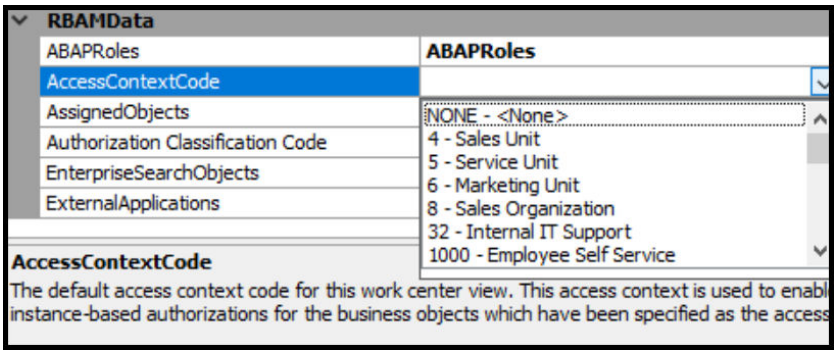
9.2.2.1 Check_0001: FindForm Consistency Check

Description	Checks if a FindFormPane has a search text binding configured and if configured, it checks the consistency of the search text/ query parameter binding.
Messages	• FindFormPane ... refers via SearchTextBinding to ..., which does not exist in the DataModel • FindFormPane ... refers via SearchTextBinding to ... and this DataField has no Query Parameter Binding • FindFormPane ... refers via SearchTextBinding to ... and its Query Parameter Binding has no ... attribute • FindFormPane ... refers via SearchTextBinding to ... and the query ... mentioned under Query Parameter Binding is not defined • FindFormPane ... refers via SearchTextBinding to ... and the query ... mentioned under Query Parameter Binding does not contain the search text parameter ...
Type	VIOLATION
Solution	Don't use the search text binding or correct the Query Parameter Binding in the Query Configuration view of the UI Designer.
Scope	OWL, FS, QA, OIF, GAF, EC, OVS, TI, MD

9.2.2.2 Check_0002: Excel Export Configuration Consistency Check

Description	Checks if the 'Group Id' is configured for Event Handlers having an operation as "DocumentOutputOperations".
Messages	In the EventHandler '...' no Group Id configured for 'DocumentOutputOperation'
Type	VIOLATION
Solution	Specify the 'Group Id' for the mentioned "DocumentOutputOperation" of the mentioned EventHandler.
Scope	OWL

9.2.2.3 Check_0003: RBAM element 'AccessContextCode' Consistency Check

Description	Checks Work Center views for its property **AccessContextCode** as it must not be empty if the **AuthorizationClassificationCode** is "ApplicationContainer". 
Messages	AccessContext attribute of CenterStructure element has no value.
Type	VIOLATION
Solution	Fill the property AccessContextCode with a suitable value. If no value applies, then fill with NONE. 
Scope	WCVIEW

9.2.2.4 Check_0004: RBAM element 'AccessCheckModeAttribute' Consistency Check

Verifies **AccessCheckModeAttribute** of RBAM Data Element when attribute **AuthorizationClassificationCode** is set as 'Application'.

Messages	Type	Corrective Actions
Property accessCheckMode is missing.	VIOLATION	**Maintain** the access check mode.

Messages	Type	Corrective Actions
Property accessCheckMode contains incorrect value: ...	VIOLATION	**Correct** the access check mode.
Property accessCheckMode is set to 'PrivilegedExceptAccessControlledBusinessObjects', but referenced BusinessObject '...' is not existing in backend	VIOLATION	**Correct** the listed access controlled object.
Property accessCheckMode is set to 'PrivilegedExceptAccessControlledBusinessObjects', but no AccessContolledBusinessObject was specified	VIOLATION	**Add** at least one access controlled object for which the extended authorization check flag is not set
Property accessCheckMode is set to 'NotPrivileged', but at least one AccessContolledBusinessObject was specified	VIOLATION	**Remove** the access controlled objects from the list
Property accessCheckMode is set to 'Privileged', but at least one AccessContolledBusinessObject was specified	VIOLATION	**Remove** the access controlled objects from the list
Property accessCheckMode is set to 'NotPrivileged'	WARNING	This value is reserved for applications for special use cases. For a normal UI application, **change** the access check mode to 'PrivilegedExceptAccessControlledBusinessObjects'
Property accessCheckMode is set to 'Privileged'	WARNING	This value is reserved for applications for special use cases. For a normal UI application, **change** the access check mode to 'PrivilegedExceptAccessControlledBusinessObjects'

Scope: OWL, FS, OIF, QA, GAF, TI, QC, QV

9.2.2.5 Check_0005: RBAM element 'AssignedObjects' Consistency Check

Description	Checks the ComponentId of **AssignedObjects** of type NavigationTarget and WocViewInclude.

Messages	• AssignedObjectType is set to 'NavigationTarget', but ComponentId does not reference an 'uicomponent' • AssignedObjectType is set to 'NavigationTarget', but ComponentId is missing • AssignedObjectType is set to 'WocViewInclude', but ComponentId does not reference an 'uiwocview' • AssignedObjectType is set to 'WocViewInclude', ComponentId is missing
Type	VIOLATION
Solution	Correct ComponentId in AssignedObject Editor. You can find the editor under *WorkCenterView Properties* > *RBAM Data* > *AssignedObjects*
Scope	WCVIEW

9.2.2.6 Check_0006: Text Category Configuration Consistency Check

Description	Checks if TextPoolEntry elements with a text length greater equal 255 characters assign their 'textCategory' attribute with the value YTXT.
Messages	• The length of the text of text pool entry for language "..." with uuid '...' is '...' (greater 120) although the text is defined with text category 'XTXT'. • The length of the text of text pool entry for language "..." with uuid '...' is '...' (greater 120) although the text is defined with text category 'XBLI'. • The length of the text of text pool entry for language "..." with uuid '...' is '...' (greater 120) although the text is defined with text category 'XLOG'. • The length of the text of text pool entry for language "..." with uuid '...' is '...' (greater 120) although the text is defined with text category 'XMSG'.
Type	VIOLATION
Solution	Reduce the length of the text.
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS, WCVIEW, PTP, QV, TI, QC, OCM, TT, OfflineQuery

9.2.2.7 Check_0007: Property 'CSN Component' Configuration Check

Description	Checks if the property CSN Component is maintained for UI Component.

Messages	• CSN component property is not selectable. Please choose a selectable one. • CSN component property is not maintained. Please add a selectable one.
Type	VIOLATION
Solution	Fill the property CSN component at the Component level property of the UI Component.
Scope	All UI Components.

9.2.2.8 Check_0008: Referenced DefaultSet Existence Check

Description	Checks the existence of DefaultSet defined for executeDefaultSet of EventHandlers.
Messages	• EventHandler '...' defines an ExecuteDefaultSet operation for a default set named '...', which is defined multiple times. • EventHandler '...' defines an ExecuteDefaultSet operation for a not defined default set named '...'.
Type	VIOLATION
Solution	Define missing default set or choose an existing default set. If a default set is defined multiple times, then rename to remove duplicates.
Scope	OWL, FS, QA, OIF, WCF, GAF, WCVIEW, EC, MD, OVS, TI, QC, QV

9.2.2.9 Check_0009: DropDownListBoxes for PostalCodes Consistency Check

Description	Checks if DropDownListBoxes for PostalCodes have set the property 'SwitchToInputOnEmptyCodes' to 'True'.
Messages	Check if the property 'SwitchToInputOnEmptyCodes' should be 'True'
Type	VIOLATION
Solution	Set the property 'SwitchToInputOnEmptyCodes' to 'True'.
Scope	OWL, OVS

9.2.2.10 Check_0010: List DataField Relative Binding Consistency Check

Description	Checks if binding of DataField in List should be relative to the association to which its parent is bound.
Messages	Binding for DataField '...' is wrong. The binding should be relative to the association to which its parent is bound.
Type	VIOLATION
Solution	Bind DataFields within a DataList relative to the association to which DataList is bound.
Scope	OWL, FS, QA, OIF, GAF, EC, OVS, TI, MD, QC, OCM

9.2.2.11 Check_0011: Hierarchical Extension Binding Consistency Check

Description

Validates if the parent and child association are bound correctly in the hierarchical extension for the given DataList:

- If the child association is not bound to a recursive association or not bound at all, then the hierarchical list does not work during runtime.
- If the parent association is not bound to a recursive association or not bound at all, although it's referenced as targetList of the locate list operation, then the hierarchical list does not work during runtime.

Message	Type	Solution
Binding of Children Association in the Hierarchical Extension for the DataList '...' is mandatory	VIOLATION	Bind the children association in this hierarchical extension for the given datalist.
Children Association in the Hierarchical Extension for the DataList '...' shall be bound to a recursive association	VIOLATION	Bind the parent association in the hierarchical extension for the given datalist.
Parent Association in the Hierarchical Extension for the DataList '...' shall be bound, since it's referenced as targetList of the locate list operation '...' for the event handler '...'	VIOLATION	Bind the parent association in the hierarchical extension for the given datalist to a self-recursive association (single association - no chain - source and target is the same BO/ECO node)

Message	Type	Solution
Parent Association in the Hierarchical Extension for the DataList '...' shall be bound to a recursive association, since it's referenced as targetList of the locate list operation '...' for the event handler '...'	VIOLATION	Bind the children association in the hierarchical extension for the given datalist to a self-recursive association (single association - no chain - source and target is the same BO/ECO node)

Scope

OWL, FS, QA, OIF, WCF, GAF, WCVIEW, EC, MD, OVS, TI, QC

9.2.2.12 Check_0012: BO Metadata Availability Check

Description

Checks whether referenced BO metadata is available.

Messages

Message	Type
'for' <referenced entity> 'in' <BO metadata reference> 'no bo metadata dom returned'	VIOLATION
<referenced entity> 'in' <BO metadata reference> 'not available'	VIOLATION
<referenced entity> 'in' <BO metadata reference> 'is no unique action'	VIOLATION
<referenced entity> 'in' <BO metadata reference> 'is a not existing action'	VIOLATION
<referenced entity> 'in' <BO metadata reference> 'is no unique association'	VIOLATION
<referenced entity> 'in' <BO metadata reference> 'is a not existing association'	VIOLATION
<referenced entity> 'in' <BO metadata reference> 'is no unique query'	VIOLATION
<referenced entity> 'in' <BO metadata reference> 'is a not existing query'	VIOLATION
'proxyName in' <BO metadata reference> 'is not defined'	VIOLATION

Message	Type
'actionPath in' <BO metadata reference> 'is not defined'	VIOLATION
'field in' <BO metadata reference> 'is not defined'	VIOLATION
'queryPath in' <BO metadata reference> 'is not defined'	VIOLATION
<referenced entity> 'in' <BO metadata reference> 'uses proxy instead of esr notation'	VIOLATION

Solution

For the violation reported, correct binding of the mentioned entity. The mentioned entity can be identified by clicking on the violation and look at the path information displayed at the bottom in the Check Results View.

Scope

OWL, FS, QA, OIF, GAF, EC, OVS, PTP, QV, TI, MD, QC, OQM

9.2.2.13 Check_0013: EventHandler Consistency Check

Description	Message	Type	Solution
Checks whether the event handler of type FireEventHandler has an event handler assigned to it.	In the EventHandler <eventHandlerName> no EventHandler configured for 'FireEventHandler'	VIOLATION	Assign an event handler to event handler of type FireEventHandler.
Checks whether the event handler of type ShowModalDialog has a Modal Dialog assigned to it.	In the EventHandler <eventHandlerName> ModalDialog not configured for 'ShowModalDialog'	VIOLATION	Assign a ModalDialog to the event handler of type ShowModalDialog.
Check if the event handler is of type DataOperation and the operation type is assign, assignText or assignConstant, then it should have source and target expression. If the operation type is clear then it should have target expression.	→ DataOperation <dataOperationName> EventHandler <eventHandler> does not have source and target expression; OR → DataOperation <dataOperationName> EventHandler <eventHandler> does not have target expression	VIOLATION	→ Assign the source and target expression if the operation is of type assign or assignText. → Assign the target expression if an operation is of type clear or assignConstant.

Description	Message	Type	Solution
Checks whether an event handler with a FirePlug element defines the required outPlug attribute.	FireOutport named <firePlugName> of EventHandler named <eventHandlerNames> does not define OutPort	VIOLATION	Define OutPort for the FireOutport of the referenced EventHandler.
Checks whether data field used in Target Expression in an Event Handler - Data Operation is present in the Data Model or not.	The target expression <fireEventHandlertargetExpression> used in highlighted Event Handler is not available in the data model	VIOLATION	Data operation in event handler cannot have a value in target expression which is not present in the data model of that UI component. This means the data element path you have used in target expression is either deleted from the Data Model or it never exists. You need to either add such data element in the Data Model or you need to change the target expression.

Scope

OWL, FS, QA, OIF, GAF, EC, OVS, TI, MD, QC, OCM, PTP, QV, TT

9.2.2.14 Check_0014: OVS Input Field Binding Consistency Check

Description	Checks if OVS Input fields have unbound data field.
Messages	Erroneous binding of <OVSElementName> OVS element detected.
Type	VIOLATION
Solution	Ensure that the OVS input field is bound to a datafield which has binding in the data model.
Scope	OWL, FS, QA, OIF, GAF, EC, OVS, TI, MD, QC, QV

9.2.2.15 Check_0015: Modeled Query Result List Binding Configuration Check

Description	Checks if modeled query result list is mapped to correct node.
Messages	Result List doesn't match mapped node.
Type	VIOLATION
Solution	Ensure that the query BO node and BO node bound to query's result list binding are the same.
Scope	OWL, FS, QA, OIF, GAF, EC, OVS, PTP, QV, TI, MD, QC, OQM

9.2.2.16 Check_0016: Query Parameter Binding Consistency Check

Description	Checks if the query parameter binding is consistent. The binding is evaluated as followed: • For each QueryParameter, the bound data element has to exist. • For each query bound DataModelElement, a referencing QueryParameter has to exist.
Messages	• Query parameter ... : binding an inconsistent, bound data model element ... does not exist. • Query bound data element ... : no related query parameter exists.
Type	VIOLATION
Solution	• Adjust query parameter binding. • Delete no longer used data elements. • Sometimes the data element exists in the data model but the property 'Query parameter Binding' is set to false. To correct this case you have to rebind the data field on the query level.
Scope	OWL, OVS, TI, EC, MD, OQM

9.2.2.17 Check_0017: MapPane Configration Consistency Check

Description	Checks if the mandatory attributes (Name, Address, and ThingTypeID) of Map Pane are configured or not.
Messages	• For referenced MapPane, No "Binding" is defined. • For referenced MapPane, Name is not bound. • For referenced MapPane, Name is not bound relative to MapPane binding. • For referenced MapPane, Address is not bound. • For referenced MapPane, Address is not bound relative to MapPane binding. • For referenced MapPane, ThingTypeID is not bound. • For referenced MapPane, ThingTypeID is not bound relative to MapPane binding.
Type	VIOLATION
Solution	The mentioned properties are mandatory and should have relative binding with respect to the Binding property of Map Pane.
Scope	OWL, FS, QA, OIF, GAF, EC, OVS, TI, MD, QC, QV, MashUp, MashupRegistry

9.2.2.18 Check_0018: BO Read Operation Consistency Check

Description	Checks Inports having OnFire events that are of BORead operation type, for each of these onFire event handlers, it performs a BO operation parameter consistency check.
Messages	Erroneous parameter binding of onFire event <eventHandlerName> detected
Type	VIOLATION
Solution	Ensure that the BO operation parameter is unbound in the data model.
Scope	OWL, FS, QA, OIF, GAF, EC, OVS, TI, MD, QC, QV

9.2.2.19 Check_0019: BOAction Operation Consistency Check

Description	Checks consistency of BOAction Operation in an EventHandler.

Messages	• BOAction ... without instanceBinding found. • BOAction ... has instanceBinding ... to a non existing DataModel element. • DataModel element ... that is referenced by the BOAction ... has no Bind element. • Binding of the DataModel element ... that is referenced by a BOAction ... is not set or empty. • BO proxyName ... of the DataModel element ... that is referenced by a BOAction ... is different than the BoAction binding ...
Type	VIOLATION
Solution	For the violation reported, entity can be identified by clicking on the violation and look at the path information displayed at the bottom in the Check Results View.
Scope	OWL, FS, QA, OIF, GAF, EC, OVS, TI, MD, QC, QV, PTP

9.2.2.20 Check_0020: OVS AutoSuggest Consistency Check

Description	Checks whether configured AutoSuggest will work for OVS or not.
Messages	No search text binding detected in FindFormPane <findFormPaneName>; OVS autosuggest will not work
Type	VIOLATION
Solution	For AutoSuggest to work in OVS, configure the SearchTextBinding parameter for the FindFormPane.
Scope	OWL, FS, QA, OIF, GAF, EC, OVS, TI, MD, QC, QV

9.2.2.21 Check_0021: DefaultSet Parameter Value Configuration Check

Description	Checks whether attribute High Value or Low Value of Query Parameters of DefaultSet is blank or null.
Messages	Attribute 'High Value' or 'Low Value' of Query Parameters of default set <default set name> is blank.
Type	VIOLATION

Solution	Ensure that the attributes ‘high value’and ‘low value’ is not left blank. If you do not need to specify any value, then enter a whitespace to indicate the same.
Scope	OWL, FS, QA, OIF, GAF, EC, OVS, PTP, TI, MD, QC, QV, OQM

9.2.2.22 Check_0022: List Column Binding Consistency Check

Description	Checks if List Column binding is valid.
Messages	ListColumn with ID <Column ID> has no binding expression
Type	VIOLATION
Solution	Go to *Properties Explorer*, and under the "Data Information" category change the "value" property and bind it to desired backend node.
Scope	OWL, FS, QA, OIF, GAF, EC, OVS, TI, MD, QC, QV

9.2.2.23 Check_0023: List Column having Explanation Consistency Check

Description	Checks if List Column with an Explanation has a Label set or not.
Messages	Label name cannot be empty if Explanation is given for ListColumn with ID <Column ID>
Type	VIOLATION
Solution	If the Explanation property under the category "Text Information" has a value, the Label property cannot be empty/null. Go to *Properties Explorer*, and under the "Text Information" category change the value of "Label" property and bind it to some non-empty value.
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC

9.2.2.24 Check_0024: File Upload Control Consistency Check

Description	Checks if File Upload Control has data structure CCTS type as BinaryObject.
Messages	Erroneous CCTS type detected for file upload control <fileUploadControlName>
Type	VIOLATION
Solution	Ensure that the CCTS type for the File Upload structure in the data model is binary.
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC

9.2.2.25 Check_0025: CIP Class Implementation Consistency Check

Description

Checks whether specified CIP class exists and whether it implements the proper message mapping interfaces.

This check verifies a UI model in the following sequence:

1. Checks whether UI model contains any message and message mapping content
2. Checks whether UseUIControler Flag is set --> If not = Error case 1
3. Checks whether Message Mappings with mapping rule: Handler exist
4. Checks whether a CIP class is specified --> If not = Error case 2
5. Checks whether CIP class exist in backend system --> If not = Error case 3
6. Checks whether the named CIP class implements the necessary message mapping interface --> If not = Error case 4
7. Checks whether CIP class does not contain a simple implementation skeleton for a break out method--> If not = Error case 5

Message	Type	Solution
Inconsistence due to the usage of Message & Message Mappings without UseUIController flag	VIOLATION	Either set UseUIController to True (Placed in: tab Designer --> Property Configuration) or remove any message & message mapping content (Placed in: tab Controller -> Messages / Message Mappings).
No CIP class specified despite the usage of Message Mapping(s) with mapping rule: Handler	VIOLATION	Enter an existing CIP class into field "CIP ClassName" (Placed in: tab DataModel -> Select Root element -> Property Design).

Message	Type	Solution
Chosen CIP class does not exist in the backend system	VIOLATION	Check-in corresponding backend system that chosen CIP class to exist and if not, create it.
Chosen CIP class does not support Message Mapping interface (if_esf_msg_text_mapping_callb)	VIOLATION	Check that CIP class supports Message Mapping interface (if_esf_msg_text_mapping_callb) and if not open CIP class in backend system and implement interface method if_esf_msg_text_mapping_callb~map_messages.
Chosen CIP class does not contain simple implementation skeleton for the break out method	VIOLATION	For usage of system message mapping break out it is at least necessary to implement an empty method body.

Scope

OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC, TaskListView, WCVIEW, WCF

9.2.2.26 Check_0026: Duplicate DataField usage in Queries Check

Description	Checks if any DataField is used more than once in different BoQueryParameter-Elements and different Queries.
Messages	DataField '...' is used more than once in Query(s): ...
Type	VIOLATION
Solution	Bind query parameters to different data fields.
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC, PTP, OQM

9.2.2.27 Check_0027: Query Attribute 'querypath' Consistency Check

Description

Check whether a query is bound to queries (querypath-attribute need to be filled).

The following checks are performed:

- Checks if the query entity contains the 'queryPath' attribute and its value is null or empty.

- Checks if the query entity contains the 'resultBinding' attribute and its value is null, empty or invalid.

Message	Type	Solution
Query <QueryName> does not contain queryPath attribute	VIOLATION	Add the valid queryPath attribute to the query mentioned.
Query <QueryName> not bound to a query path	VIOLATION	Fill the correct value for queryPath attribute.
Query <QueryName> does not contain resultListBinding attribute	VIOLATION	Add the valid resultListBinding attribute to the query mentioned.
Query <QueryName> not bound to a resultListBinding	VIOLATION	Fill the correct value for resultListBinding attribute.
Query <QueryName> has an invalid binding for resultListBinding	VIOLATION	Correct value for resultListBinding attribute.
Query <QueryName> has to be bound to a query under the association <AssociationName>	VIOLATION	Bind the query to a query under the given association.

Scope

OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC, PTP, OQM

9.2.2.28 Check_0028: BO Query Parameter Consistency Check

Description	Checks if the query parameters of a query are bound to a data field or has a constant value.
Messages	Query parameter ... is not bound.
Type	VIOLATION
Solution	Bind the query parameter to a data field.
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC, PTP, OQM

9.2.2.29 Check_0029: Query Parameter Binding Consistency Check

Description	Checks whether data fields used in query parameter value should not be bound to BO Model.
Messages	Query parameter value <parameter bind value> should not be bound in Data Model
Type	VIOLATION
Solution	Use only those data field binding in query parameter values which are not bound to BO node in BO browser. If a query parameter data field is bound to a BO node then it may lead to runtime backend dumps.
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC, PTP, OQM

9.2.2.30 Check_0030: BO Operation Binding Consistency Check

Description	Checks whether BO Operation has valid binding. It checks if the event handler is of type 'BO Operation' and operation type is 'Read', then the parameters of the BO Operation should have binding.
Messages	Parameters of BO Operation EventHandler ....does not have binding.
Type	VIOLATION
Solution	Bind the parameters of event handler of type BO Operation.
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC, PTP

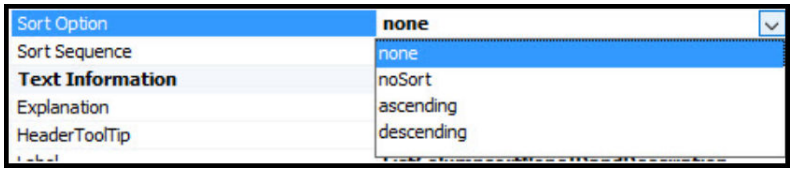
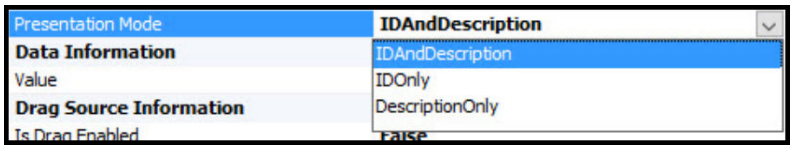
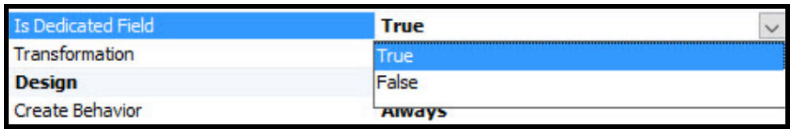
9.2.2.31 Check_0031: CodeList Configuration Consistency Check

Description	Checks whether following attributes are maintained for following Codelists: • Static and Dynamic Codelist - *typeName*, *esrNamespace* • ListBound Codelist - *listBinding*, *codeField*, *valueField*

Messages	• typeName or esrNamespace is empty, but need to be defined on Codelist. • typeName or esrNamespace is missing, but need to be defined on Codelist. • List Binding, List Code Field, and Value Code Field properties need to be maintained for a List Bound CodeList.
Type	VIOLATION
Solution	Bind field if not bound and run Metadata Update in case of Static and Dynamic Codelist. In case of listBound Codelist maintain listBinding, listCodeField, and listValueField properties.
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC, TT

9.2.2.32 Check_0032: Sort Configuration on a List Column Check

Description	Checks if any list column has "Sort" option set and has a dedicated field binding then sorting is not possible.
Messages	Sorting is not possible on highlighted List Column.
Type	VIOLATION

Solution	Ensure that List Column does not have a dedicated field or ensure that the value for isDedicatedField is false in a dedicated data field. Possible corrections are as folows: • List Column has four values for Sort Option. The value can be set to noSort.  • List Column has three values for presentation mode. The value can be set to IDOnly.  • Dedicated Field has a boolean attribute, Is Dedicated Field. The value can be set to False. 
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC

9.2.2.33 Check_0033: RubyScript to JavaScript Conversion Possibility Check

Description	Checks the Ruby Scripts in the UIComponent for any inconsistency which might prevent successful conversion to Javascript. Correct javascript conversion is essential for consumption by Runtime client.
Messages	See Table: Message List
Solution	(1) Correct the inconsistency of the script as mentioned in the message . (2) Add script or remove script operation in case it's not needed.
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC, WCF, TT

Message List

	Message	Type	Examples
1	Error in converting RUBY script to JS (Reason unknown / no regex to detect)	VIOLATION	N/A
2	Keyword var cannot be used in script	VIOLATION	var a;
3	Keyword document cannot be used in script	VIOLATION	var document;
4	Keyword synchronized cannot be used in script	VIOLATION	var synchronized;
5	Keyword this cannot be used in script	VIOLATION	var this;
6	Keyword enum cannot be used in script	VIOLATION	var enum;
7	Keyword window cannot be used in script	VIOLATION	var window;
8	There should be no space between - symbol and numeric value when used after a comparison operator	VIOLATION	if(a !== - 8)
9	Do not use assignment operator inside condition expression	VIOLATION	if(a=8) -
10	There should be space between != and first operand	VIOLATION	a!=8
11	Do not leave an empty else block	VIOLATION	else end
12	Do not use Multi-line comments, they are not supported	VIOLATION	/* comments like this is not allowed */
13	if! and elseif! is not supported. Please use a space between if or elseif and !	VIOLATION	if!(a == 8)
14	Cannot have space before semi colon of last line(only reactivation required)	VIOLATION	
15	Increment and decrement operator not supported	VIOLATION	a++ or a--
16	elseif block cannot be empty	VIOLATION	elseif(a == 8) end
17	String or value between concatination operators missing	VIOLATION	a = "Value(s) is missing" + + " in between";
18	Incorrect usage of operators	VIOLATION	a += 8;
19	No delimiter after assignment operation use; to delimit statements	VIOLATION	

	Message	Type	Examples
20	Should not use keyword "_createLeadSelectedIndex" in script	VIOLATION	
21	Should not use keyword "_getRowByID" in script	VIOLATION	
22	Should not use keyword "_getFormattedValue" in script	VIOLATION	
23	Should not use keyword "_getDateTimeValue" in script	VIOLATION	
24	Should not use keyword "_getDataRow" in script	VIOLATION	
25	Should not use keyword "_getLeadSelectedIndex" in script	VIOLATION	
26	Should not use keyword "add" in script	VIOLATION	
27	Should not use keyword "clear" in script	VIOLATION	
28	Should not use keyword "DataWrapper" in script	VIOLATION	
29	Should not use keyword "delete" in script	VIOLATION	
30	Should not use keyword "Get" in script	VIOLATION	
31	Should not use keyword "GetVaue" in script	VIOLATION	
32	Should not use keyword "SetValue" in script	VIOLATION	
32	Should not use keyword "SetLeadSelectedRowIdentifier" in script	VIOLATION	
34	Should not use keyword "SetLeadSelectedIndex" in script	VIOLATION	
35	Should not use keyword "getCctsType" in script	VIOLATION	
36	Should not use keyword "GetSelectedRow" in script	VIOLATION	

9.2.2.34 Check_0034: List Operation Rule Check

Description	Checks if there is a valid target list assignemnt, for all Event Handlers of type **List**.
Messages	• List operation <&1> has got no target list assignment. • Assigned target list <&1> of list operation <&2> is not valid.
Type	VIOLATION
Solution	Choose a target list from the drop down:
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS

9.2.2.35 Check_0035: Message Mapping Consistency Check

Description	Verifies the consistency of all existing message mappings.
Messages	Inconsistence in a mapping of rule "Replace" due to missing target message information.
Type	VIOLATION
Solution	1. Check if a message is created in the *System Messages* section. 2. If not, select a target message from the *New Message Text* field .
Scope	OWL, FS, QA, WCF, GAF, EC, MD, OVS

9.2.2.36 Check_0036: Event Handler Controller Operation Check

Description	Checks the consistency of an Event Handler of the controller operation type.
Message	Inconsistency in controller operation due to missing information.
Type	VIOLATION
Solution	Enter details in the *Action Instance Binding* field. This filed is mandatory.

Scope	OWL, FS, QA, WCF, GAF, EC, MD, OVS

9.2.2.37 Check_0037: Advanced List Pane Check

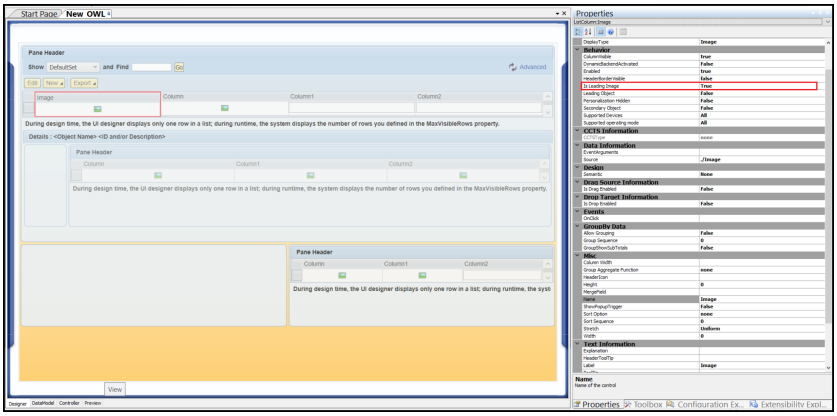
Description	Checks if UI models contain more than one Query/DefaultSet-based AdvancedListPane each with missing Active DefaultSet Binding definition. This leads to a wrong runtime behavior, as implicit some defaultsetbinding-DataField is created (only one) which is used than concurrently by the defaultset-based ALPPs..
Message	The referenced Query/DefaultSet-based AdvancedListPane defines no Active DefaultSet Binding.
Type	VIOLATION
Solution	If there are two or more contained AdvancedListPanes in one UI, you must model for each AdvancedListPane a separate DataField containing the name of the current DefaultSet, which is bound in the Active DefaultSet Binding for each AdvancedListPane (So, for each Query/DefaultSet-based AdvancedListPane an Active DefaultSet Binding is needed, which is bound to a separate DataField, which returns the current DefaultSet name). This is required because by a missing Active DefaultSet Binding, a binding to one DataField will automatically created. In case of 2 or more contained AdvancedListPanes in one UI with missing Active Defaultset Binding, this implicit created defaultsetbinding-DataField will be used than concurrently by all these defaultset-based ALPPs with missing Active Defaultset Binding.
Scope	OWL, FS, OIF, QA, WCF, GAF, WCVIEW, EC, MD, TI, OVS

9.2.2.38 Check_0038: Reference Anchor Deletion Consistency Check

Description	Checks if any anchor used as a Reference Anchor has been deleted from UI component.
Message	Referenced Anchor <ReferencedAnchorName> does not exist in UIComponent.
Type	VIOLATION

Solution	You cannot have an anchor used as a reference anchor in any Inport or Outport if that anchor does not exist in the UI component. Either undo the deletion of a specified anchor, or remove Referenced Anchor from Inport/Outport from the UI Component.
Scope	OWL, FS, OIF, QA, WCF, GAF, WCVIEW, EC, MD, OVS, TaskListView

9.2.2.39 Check_0039: Advanced List Column Selection Check

Description	Checks if advanced list has more than one column selected as leading image.
Message	Advanced list Pane <advanced list pane ID> has more than one column selected as Leading Image.
Type	VIOLATION
Solution	For any list at column level there is a property to set the column as Leading Image. This property can be set to true only for one column for every advanced list. So if more than one column is set as leading image, set the value of the property *isLeadingImage* to **false** for all the columns except the one that is the leading image for that list. 
Scope	All UI components where you can add a list.

9.2.2.40 Check_0040: Query Parameter in Offline Query Model Check

Description	Checks if the data field used as Query Parameter of Offline Query Model (OQM) has Offline binding (OCM Binding) or not.
Message	OCM Binding is missing for Query Parameter DataField <data field Name>.
Type	VIOLATION
Solution	It is mandatory to have all the data fields which are used as Query Parameters in OQM to have OCM binding. This is required for fetching Offline Queries in PDI ABSL.
Scope	Offline Query Model (OQM)

9.2.2.41 Check_0041: Static Text Controls Check

Description	Checks if the TextEdit or the StaticText controls in OWL component have *SupportFormattingText* property set to **True**.
Message	<Control Name> with ID <Control ID> has attribute *SupportFormattingText*set to true.
Type	VIOLATION
Solution	The TextEdit or the StaticText controls in an OWL Component cannot have the *SupportFormattingText* property set to **True**. Setting this property to true converts these controls to Rich Text Editor. This dumps on UI as formatting is not supported for Rich Text Editor. Change this property to false to solve the error. To do this, go to *Properties Explorer* and under the Behavior category change the value of *SupportFormattingText* property to **False**.
Scope	All OWL Components.

9.2.2.42 Check_0042: Mass Copy Paste Property Check

Description	If the property *EnableMassCopy-Paste* for a particular list is set to **true**, then that list should not be: • Hierarchical • Unbound • Used in Query ResultListBinding
Messages	• The query <Query name> cannot have the same result list binding as that of list as*EnableMassCopy-Paste* property is set to **true**. • List <List name> with ID <List ID> cannot be a hierarchical list as *EnableMassCopy-Paste* property is set to **true**. • List <List name> with ID <List ID> cannot be unbound as *EnableMassCopy-Paste* property is set to **true**.
Type	VIOLATION
Solution	• Set the *EnableMassCopy-Paste* property to **false**; OR • Bind the Datalist; OR • Delete the Hierarchical List from Datalist; OR • Do not use the list as resultListBinding in any query.
Scope	OWL, FS, OIF, QA, GAF, EC, MD, OVS, QV, TI, QC

9.3 Modeling Floorplans

9.3.1 Floorplan Modeling Quick Guide

This document gives you an overview of the steps involved in modeling different types of floorplans in the studio using the user interface designer.

In the studio, you can generate screens for the business objects in your solution.

For more information, see Generate Screens for a Business Object [page 429].

Business and technical background

SAP's User Interface Style Guide

You can learn how to build user interfaces according to this style guide.

Tasks

Model a Fact Sheet

You can create a fact sheet to display a read only overview of an object.

For more information, see here [page 697].

Model a Quick Activity Floorplan

You can create a quick activity to display an overview of a business object.

For more information, see here [page 700].

Model an Object Instance Floorplan

An object instance floorplan (OIF) is used to create and edit business object data or trigger actions on a business object. It is opened using object-based navigation. It can be used to create new business object data. OIFs are similar to quick activities except that an OIF has a set of tabs such as General and Line Items.

For more information, see here [page 702].

Model an Object Work List

You can create a list pane that is always embedded in a work center view. The OWL contains business objects such as purchase orders or opportunities. You can, for example, search for, display, and edit business objects as well as trigger follow-on processes.

For more information, see here [page 703].

Create an Object Value Selector Dialog Box

You can create an object value selector (OVS) dialog box as a reusable component. It can be called from any OVS control.

An OVS helps you select one item from a table of business object instances.

For more information, see here [page 705].

Model a Guided Activity Floorplan

You can create a guided activity floorplan to simplify complex tasks. This floorplan helps you navigate step by step through a given process.

For more information, see here [page 708].

Model a Work Center View

You can model a work center view to hold a list of related floorplans. The work center view is a component that can be used across different work center floorplans that allow easy navigation to different sections from within the specified floorplan.

For more information, see here [page 710].

Model a Work Center

You can model a work center floorplan to display the various work center views.

For more information, see here [page 712].

Create a Port Type Package

You can create a port type package in the studio and modify its attributes in the user interface designer. A port type package is a component that hosts one or more port types. A port type is a blueprint for an inport or an outport. Navigation between components that have inports and outports within the same port type package can be facilitated without any mapping effort.

For more information, see here [page 713].

Model a Modal Dialog

You can model a modal dialog to create a dialog box that is triggered by an event action. You can model a reusable modal dialog and then bind it to a floorplan, or you can model a modal dialog that is specific to a floorplan and that can be called by an event handler.

For more information, see here [page 714].

Model an Embedded Component

Embedded components are reusable subscreens that facilitate:

- Inport and outport navigation between the parent component and the embedded component.
- Bindings to reference data from the parent component.
- Configuration to let the parent component influence the behavior of an embedded component.

For more information, see here [page 716].

Create a Business Configuration View

You can create a business configuration view (BC view) for a business configuration set (BC set) to allow administrators to add new values during fine-tuning or to delete the values of a BC set. You create the view in the studio and you model the view in the UI designer.

> **i** Note
>
> You can only create a BC view for a business configuration set (BC set) that was created using a custom business configuration object (BCO). You cannot create a BC view for an SAP BCO.

For more information, see here [page 389].

Save and Activate Your Changes

After you have created any of the floorplans listed above, you have to save and activate the floorplans.

1. From the *File* menu, choose *Save*.
2. From the *File* menu, choose *Activate*.
3. In the *Activate Worklist* dialog box, select the check boxes beside the relevant components and click *OK*.

9.3.2 Tasks

9.3.2.1 Model a Fact Sheet

Overview

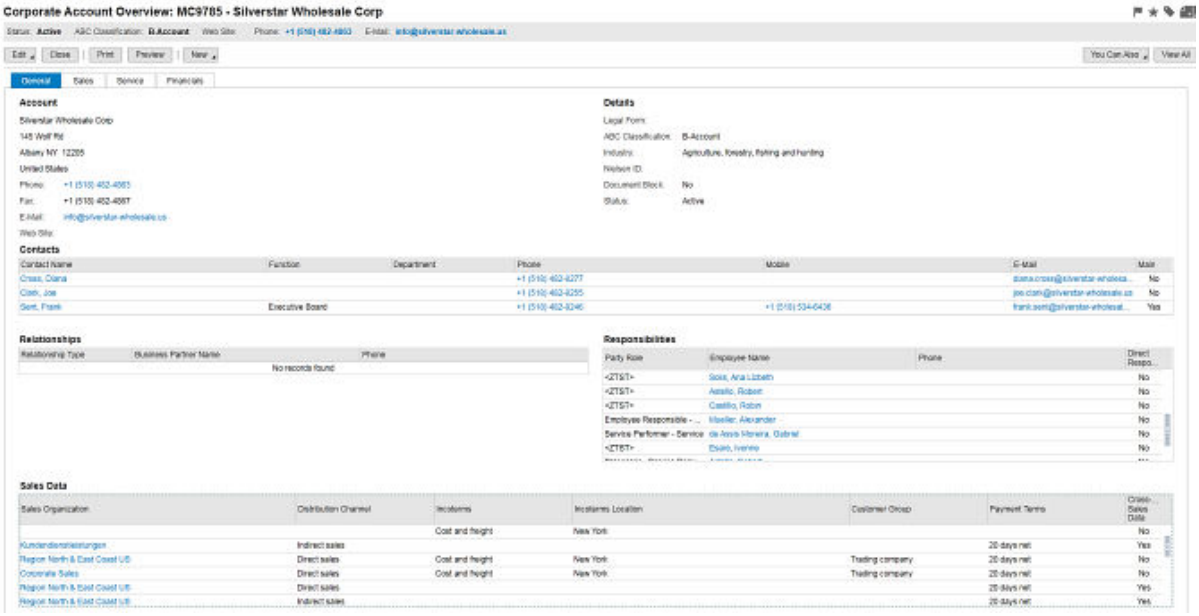
You can create a fact sheet to display a read only overview of an object. It displays the object properties, can have different views, and can contain links for navigating to related objects.

For example, an *Account Overview* fact sheet can have the sections *Accounts*, *Sales Data* and *Relationships*

If you create an *Employee* fact sheet, employees can view their data but cannot change any information.

If you want to allow users to edit the screen, you must create a quick activity floorplan or an object instance floorplan.

Here is a sample fact sheet



Fact Sheet

Prerequisites

- You have created a solution in the studio.
  For more information, see , create a Solution [page 177]
- All the required application entities such as the business object and the embedded component have been modeled in the solution.
  For more information, see Business Object Modeling [page 81].
- You have the detailed information needed for the fact sheet that you are creating.

Procedure

1. **Create a Fact Sheet**
   1. In the *Solution Explorer*, expand the solution to display the business object.
   2. Right-click on the business object and select *Create Screens*.
   3. In the *Create New Screen* dialog box, do the following:
      1. In the *Screen Name* field, enter a name for the fact sheet. For example, **Account Overview**
      2. Under *FloorplanType*, select *Fact Sheet (FS)*.
      3. Click *OK*.
         The **Account Overview** fact sheet is created and displayed in the *Solution Explorer* under the business object.
2. **Add the required UI elements and change field names**

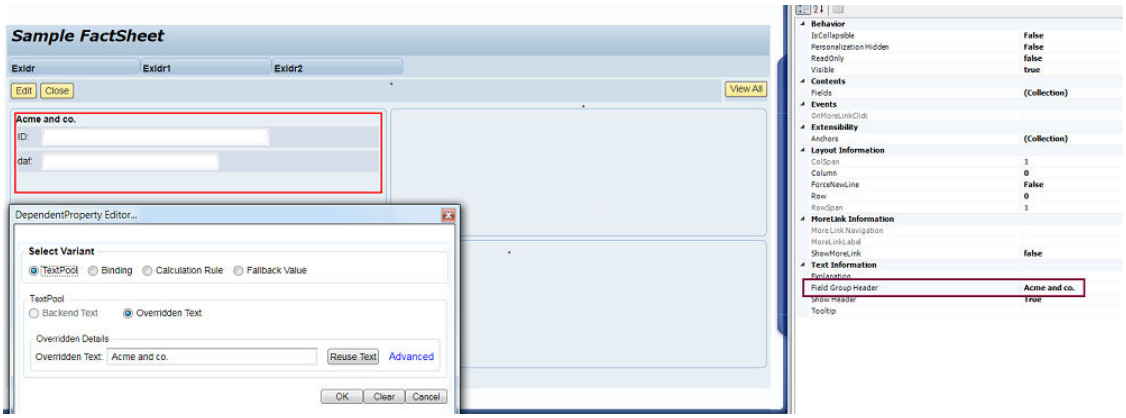
1. Double-click the fact sheet that you just created.
   The fact sheet opens in the user interface designer with the default UI elements.

> **i Note**
>
> If required, you can change the component type from **`Fact Sheet (FS)`** to **`Quick Activity (QA)`** in the *Properties Explorer*, under *Component Info-Component Type*.

2. Select the identification region
3. Enter a title for the fact sheet in the *Properties*, under **`Misc – FloorplanTitle`**
4. Drag the required controls from the *Toolbox* to the fact sheet.
   For example, you can add the fields required for general information, contacts, relationships and so on.

Drag and drop the required controls

5. Do the following for **each** of the required static text fields that are available in the **Account Overview** fact sheet.
   1. Select the static text field.
   2. In the *Properties Explorer*, under *Data Information – Text*, click the arrow button.
   3. In the *Dependent Property Editor*, select the required element and click *OK*.
   4. If you do not want to use the associated back end text for a static text field, enter a name for the field in the *Properties Explorer*, under *Text Information – Field Group Header*.



3. **Model Elements**

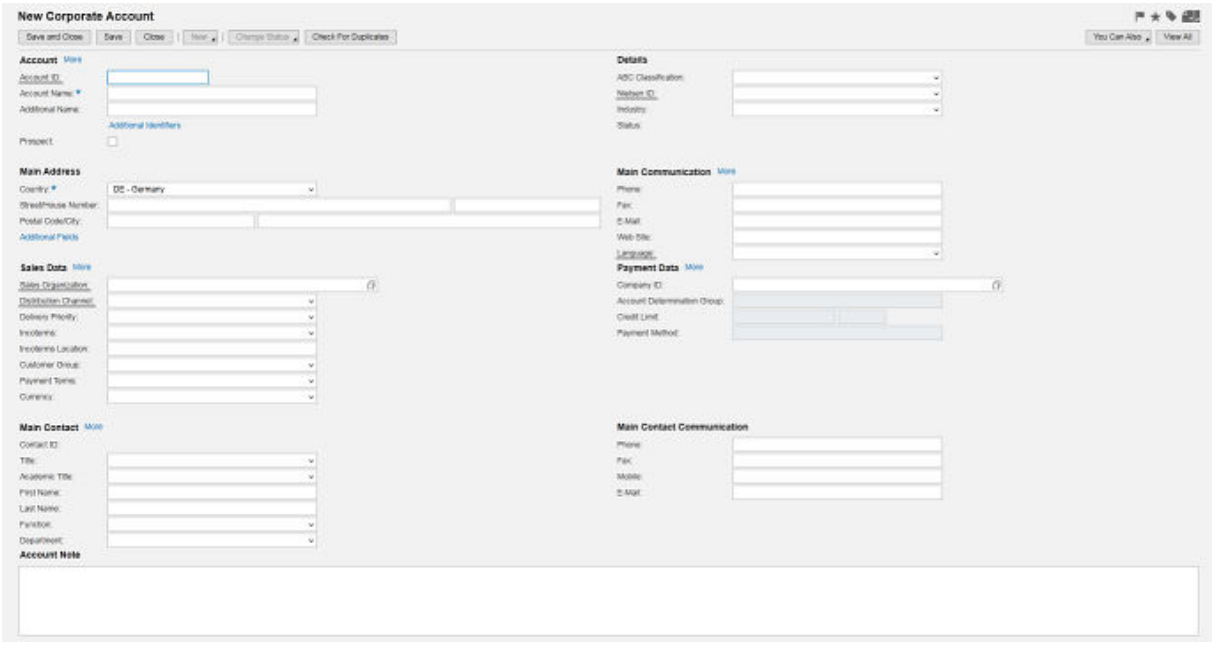
1. In the *Data Model* tab, all the elements of the selected business object are displayed. Bind each of the data model elements to a business object element.
   For more information, see Bind Data Elements to Business Object Elements [page 731].
2. Create and configure the required event handlers in the *Controller* tab. For example, create and configure event handlers for each of the buttons available on the fact sheet.
   For more information, see Create an Event Handler [page 747].
3. If you want to provide value help for any of the fields, see Configure an Object Value Selector [page 749].
4. To enable the fact sheet to be opened using object-based navigation, configure the fact sheet so that it can receive parameters from any floorplan.
   For more information, see Configure an Object-Based Navigation in a Target Floorplan [page 742].
5. Save and activate your changes.

9.3.2.2 Model a Quick Activity Floorplan

Overview

You can create a Quick Activity Floorplan (QAF) to display an overview of a business object. You can further view all the details of the business object in the associated object instance floorplan.

Here is an example of a sample QAF



Quick Activity Floorplan

Prerequisites

- You have created the required business object.

Procedure

1. **Create a quick activity**
   Let's create a QAF called *New Corporate Account*:
   1. In the *Solution Explorer*, expand the solution to display the business object.
   2. Right-click on the business object and select *Create Screens*.
   3. In the *Create New Screen* dialog box, do the following:
      1. In the *Screen Name* field, enter *New Corporate Account* as the name for the quick activity.
      2. Under *FloorplanType* select *Quick Activity floorplan (QAF)*.
      3. Click *OK*.
         The quick activity is created and displayed in the *Solution Explorer* under the business object.
   4. Double-click the new QAF.
   5. The QAF opens in the user interface designer. It has the following attributes:
      - An identification region with the title of the quick activity
      - A contextual navigation region with default buttons such as *Close*, *Save and Close*, and *View All*
      - An event handler with the operation type *WindowAction* named `Close` associated with the *Close* button
2. **Edit the quick activity**
   1. Enter a title for the quick activity in the *Properties Explorer*, under *Misc – FloorplanTitle*.

      > **i Note**
      >
      > If this floorplan allows you to either create or edit details, use a calculation script to dynamically render the title.
      >
      > For more information, see Define Dynamic Behavior of a Field [page 732] and Write a Script [page 746].

   2. **Model the quick activity**
      1. Configure the *View All* button to navigate to the corresponding **object instance floorplan** to view the details of the object.
         For more information, see, Configure an Object-Based Navigation in a Source Floorplan [page 744].
      2. If required, add additional buttons and assign an event handler to the *OnClick* event.
         For more information, see Configure a Button [page 733] and Create an Event Handler [page 747].

         > **i Note**
         >
         > For BO actions, use the operation type *BOAction* and bind the action to the BO action. The fields *Action Type* and *Instance Binding* will be automatically updated as per the BO action property.

      3. Drag the required controls from the *Toolbox* to the quick activity.
      4. In the *Data Model* tab, bind each of the data model elements to a business object element.
         For more information, see Bind Data Elements to Business Object Elements [page 731].
      5. Create an OBN inport to receive the navigation key from the source floorplan, in order to retrieve data from the target floorplan:
         1. In the *Controller* tab, right-click on *Inports* and choose *Add Inport*.

2. Check *OBN Inport*.
3. Select an operation, *PortType Package* and a *PortType Reference* as **`OpenByKey`**.
4. Add the parameter *Key List* and *Key* and bind the parameter to an unbound data list and data field that acts as a placeholder for the key.
5. Create an event handler as follows:
    1. In the *Operation Type* field, select **`BOOperation`**.
    2. In the *BO Operation Type* select **`Read`**.
    3. In the *Navigation Type* select **`Key Navigation`**.
    4. Bind the previously used, unbound data field in the ... column.
6. In the *Properties Explorer* for the newly created inport, under *Events — OnFire*, select the even handler you have created.

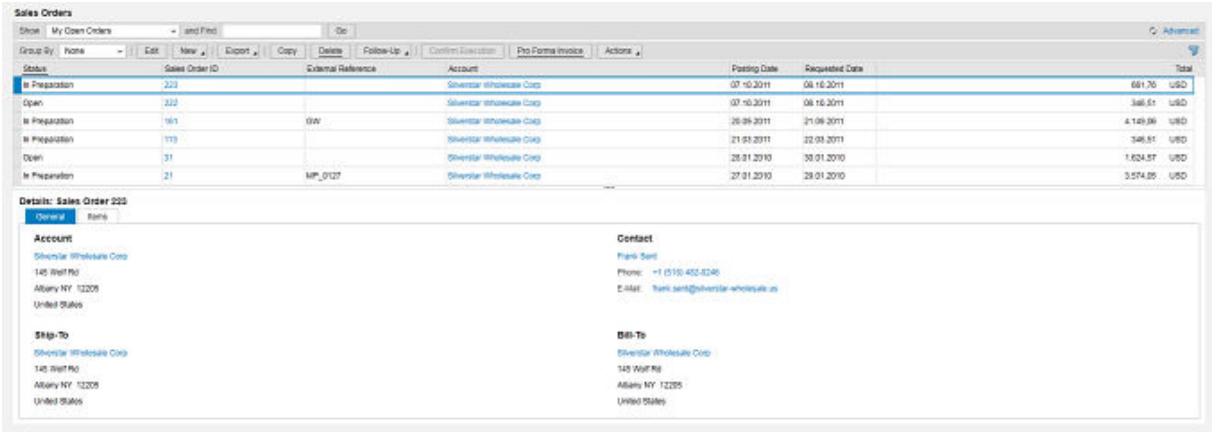
6. Save and activate your changes.

9.3.2.3 Model an Object Instance Floorplan

Overview

An object instance floorplan (OIF) is used to create and edit business object data or trigger actions on a business object. It is opened using object-based navigation. It can be used to create new business object data. OIFs are similar to quick activities except that an OIF has a set of tabs such as General and Line Items.

Here's a sample OIF:



Object Instance Floorplan

Procedure

1. **Create an OIF**
    1. In the *Solution Explorer*, expand the solution to display the business object.
    2. Right-click on the business object and select *Create Screens*.
    3. In the *Create New Screen* dialog box, do the following:
        1. In the *Screen Name* field, enter a name for the object instance floorplan.

2. Under *FloorplanType* select *Object Instance Floorplan (OIF)*.
3. Click *OK*
   A new object instance floorplan is created and displayed in the *Solution Explorer* under the business object.

2. **Model the OIF**
   1. Double-click the new OIF.
      The OIF opens in the user interface designer.
   2. In the *Designer* tab, select an OIF tab, for example, *General Data* tab.
   3. In the *Properties Explorer*, select *ViewSwitchNavigation* from the drop-down list and under *Misc-NavigationItems*, click *...*.
      In the *OIF View Collection Editor*, remove any unwanted tabs.
      Edit the properties of each of the tabs. For example, the title of the tab, its behavior, and visibility.

      > **i** Note
      >
      > The name of the tab is stored in run time in the data model field /Root/UIState/ViewSwitchSelectedItem. You can use it in dynamic data or property handling for a calculation rule.

   4. If you want to add sub tabs to a tab, do the following:
      1. In the *Properties Explorer*, select *ViewSwitchNavigationItem:NavigationItem* and under *Misc-ChildNavigationItems* click *...* .
      2. In the *OIF SubView Collection Editor*, click *Add*.
      3. Edit the properties of the sub tab, such as the title.
   5. Design each of the tabs and sub tabs, and configure the OIF similar to a quick activity.
      For more information, see steps 7 to 11 of Model a Quick Activity Floorplan [page 700].
   6. Save and activate your changes.

9.3.2.4 Model an Object Work List

Overview

You can create a list pane that is always embedded in a work center view. The OWL contains business objects such as purchase orders or opportunities. You can, for example, search for, display, and edit business objects as well as trigger follow-on processes.

Prerequisites

You have created an object instance floorplan (OIF), or a fact sheet.

For more information, see Model an Object Instance Floorplan [page 702] or Model a Fact Sheet [page 697].

Steps

1. Create an object work list (OWL)

1. In the *Solution Explorer*, expand the solution to display the business object.
2. Right-click on the business object and select *Create Screens*.
3. In the *Create New Screen* dialog box, do the following:
   - In the *Screen Name* field, enter a name for the object work list.
   - Under *FloorplanType* select *Object Work List (OWL)*.
   - Click *OK*.
     A new object work list is created and displayed in the *Solution Explorer* under the business object.
4. Double-click the new OWL.
   The OWL opens in the user interface designer.

2. Configure the data model view of the OWL

1. Add columns in the OWL by dragging and dropping *Controls* from the *Toolbox*.
2. Bind each of the fields to the required business object element.
   For more information, see Bind Data Elements to Business Object Elements [page 731].

3. Configure the design view of the OWL

On the *Designer* tab, you can see that the following fields are available by default:

- List or advanced list pane with columns
- *Show* drop-down list, basic *Find* field, *Go* button in the Search area

1. In the *Properties Explorer*, change the properties of the required controls.
2. In the *Properties Explorer*, enable the preview area by selecting *AdvancedListPaneVariant*.
   Under *Appearance-UsePreviewPane*, select `True`.
3. In the *BO Browser/Data Model*, add the necessary fields from the *DataList*.
4. To add the search parameter fields in the find form area, do the following:
   1. Click *Advanced*.
   2. In the *BO Browser* tool window set a filter to show queries.
   3. Drag query parameters to the find form.

   If you add the query parameters like this, then you do not need to add the query parameters and do the binding in the *Controller* tab.
5. To configure a button to open target floor plan from the OWL, for example, an OIF or a GAF do the following:
   1. Add an *Edit* button to the tool bar
      For more information, see Configure a Button [page 733].

2. Select the *Navigation* property and open the editor.
3. Select the namespace, OIF/FS controller, and its inport operation *Open* and click *OK*.
4. An OBN, an outport, and an event handler are created automatically in the *Controller* tab.

4. Configure the controller tab of the OWL

1. In the *Controller* tab there is a default query. You can add more queries and default sets using the context menu.
2. In the *Designer* tab, assign a title to the default set that is displayed in the *Show* drop down list .
3. In the *Controller* tab, bind the query to the controller's query using the elements structure.
4. Add and bind the query parameters from the controller to *Search Parameter Structure* fields.
5. Find the eventhandler, outport, and OBN that were created while configuring the *Edit* button.
6. To configure the outport, add the parameters *Key List* and *Key* and bind the parameter to an unbound data list and data field that acts as a placeholder for the key.
7. Save and activate your changes.

Related Information

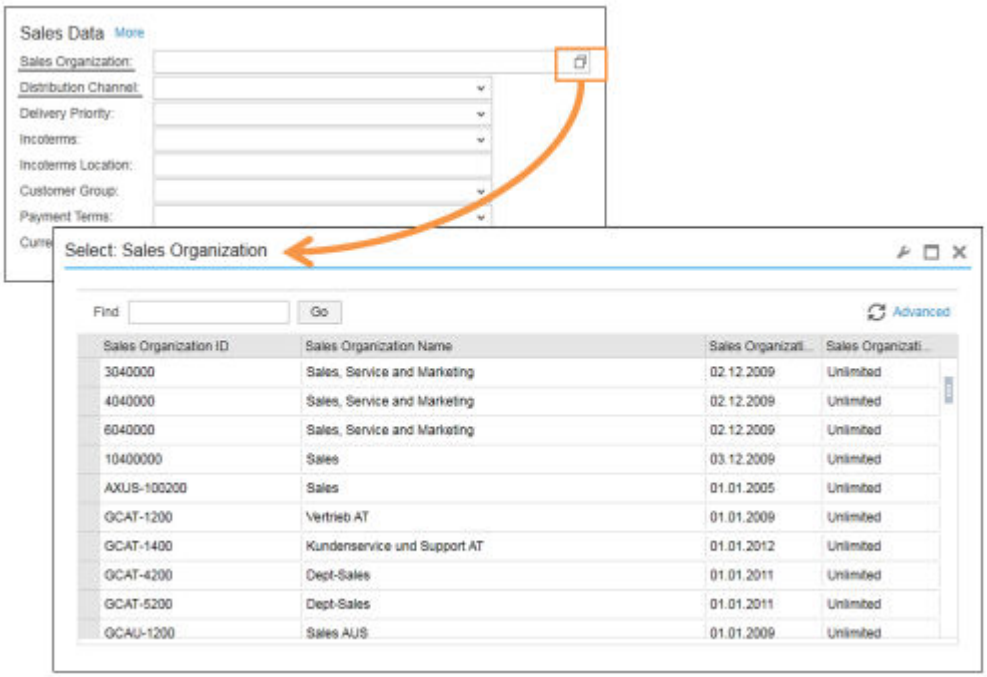
Create an OWL Excel Export [page 431]

9.3.2.5 Create an Object Value Selector Dialog Box

Overview

You can create an object value selector (OVS) dialog box as a reusable component for a custom business object and for an SAP business object. It can be called from any OVS control.

An OVS helps you select one item from a table of business object instances. This is an example of an object value selector dialog box in SAP Business ByDesign:



Object Value Selector Dialog Box

Prerequisite

You have opened a solution. If you want to add the object value selector dialog box to a custom floorplan, you need to have created a business object.

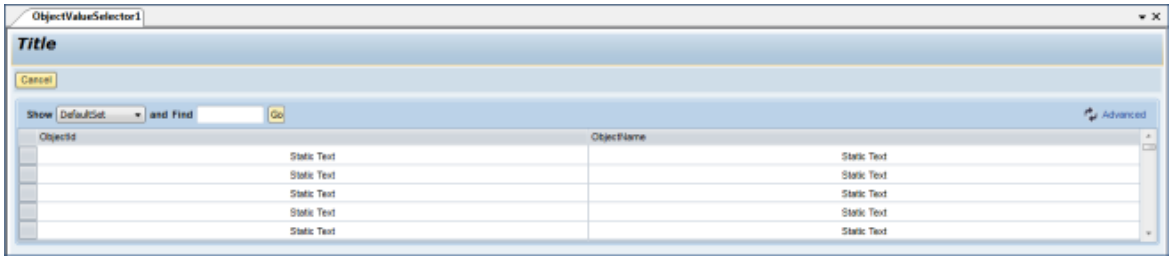
Procedure

1. In the *Solution Explorer*, do the following:

Custom Business Object	SAP Business Object

1. Right-click the business object and select *Create Screens*.
2. In the *Create Screens* dialog box, do the following:
  1. In the *Screen Name* field, enter a name for the OVS dialog box.
  2. Select the required option from the *Supported Devices* value help.
  3. Under *Screen Type*, select *Object Value Selector (OVS)*.
3. In the *Select Business Object Elements for OVS* dialog box, select the business object element(s) and click *OK*.
A new OVS dialog box is created and displayed in the *Solution Explorer* under the business object (`[Name of the business object]_OVS.OVS.uicomponent`,).
	1. Right-click the solution and select *Add New Item*.
	The *Add New Item* dialog opens.
	2. Select *SAP* > *Screens* > *Object Value Selector (OVS)* > and click *Add*.
	The *Select Business Object* dialog opens.
	3. Select an SAP business object or a custom business object or search for a business object.
	4. Optional: Change the default name.
	5. Click *OK*.
	A new OVS dialog box is created and displayed in the *Solution Explorer* under the business object (`[Name of the OVS].OVS.uicomponent`,).

2. Double-click the new OVS.
   The OVS opens in the user interface designer. A basic object value selector dialog box for an SAP business object looks as follows:



Object Value Selector for an SAP Business Object

   You now need to define the following attributes:
   - Event handlers: Used to close the window, to retrieve the value set, to handle the selection change and to return the selected value
   - Query to display the queried business data
   - Inport: Used to access the OVS
   - Outport: Used to transfer the required data from the OVS to the floorplan where it is being used.
3. In the Property Explorer, select identification region in the OVS dialog.
4. In the *Properties Explorer*, under *Misc-Window Title*, click the arrow button.
5. In the *Dependent Property Editor*, enter the title and click *OK*.
6. In the *Data Model* tab, bind the following:
   - Root to the corresponding OVS (enhanced) controller object

- List to the ECO node
- List attributes to the ECO node attributes

For more information, see Bind Data Elements to Business Object Elements [page 731].

7. In the *Controller* tab do the following:
   - Select the event handlers *GetValueHelpHandler* and *SetValueHelpHandler* and define the appropriate business object model.
   - Select the query of the data model and bind it to the business object.
   - Select the outport *OVSValueSelectorOutPort* to define the correct parameter binding from the parameter Object ID and Object Name to the attributes of the data model.
8. Save and activate your changes.

Follow-On Processes or Tasks

After you have created an OVS dialog box, you can create an OVS control. For more information, see Configure an Object Value Selector [page 749].

9.3.2.6 Model a Guided Activity Floorplan

Overview

You can create a guided activity floorplan to simplify complex tasks. This floorplan helps you navigate step by step through a given process.

Procedure

1. **Create a guided activity.**
   1. In the *Solution Explorer*, right-click your project and select *Add* > *New Item*.
      The *Add New Item* dialog box opens.
   2. Select *Guided Activity Floorplan (GAF)*.
   3. In the *Name* field, enter a name for the guided activity floorplan and click *Add*.
      The guided activity floorplan is added to your project.
   4. Double-click the guided activity floorplan.
      The guided activity floorplan opens in the user interface designer.
   5. A new guided activity shows the following:
      - An IdentificationRegion with the title of the floorplan
      - A RoadmapNavigation region with standard steps such as *Step*, *Review* and *Confirmation*
      - A ComponentToolbar with default buttons such as *Previous*, *Next*, *Finish*, and *Cancel*.
2. **Design the layout and bind the data elements**.

1. Rename each of the steps:
    1. On the *Designer* tab, select a step in the road map navigation region.
    2. In the *Properties Explorer*, under *Design — Title*, click the arrow button.
       The *Dependent Property Editor* opens.
    3. In the *Overridden Text* field, enter the name for the step.
    4. Click *OK*.
    5. Under *Misc — Name*, enter the name of the step.
       The name of the current active tab is stored in the data model field /Root/UIState/ViewSwitchSelectedItem. You can use it in dynamic data or property handling for calculation rules.
2. If required add additional steps by right-clicking on the RoadmapNavigation region.
3. Drag the required controls from the *Toolbox* to each step in the guided activity.
4. In the *Data Model* tab, bind each of the data model elements to a business object element.
   For more information, see Bind Data Elements to Business Object Elements [page 731].

3. **Configure the navigation buttons**.
    1. In the *Controller* tab, create two event handlers with operation type *Script*, say `HandlePrev` and `HandleNext`.
    2. Write a ruby script such that when the event handlers are called, they allow navigation to the next step.
    3. Under *Scope Fields* select a data field as an invalidation trigger, such that whenever the value of the field changes the script is executed.
    4. Assign the event handlers to the *OnClick* event of the *Previous* and *Next* buttons respectively
    5. To dynamically enable and display the *Previous*, *Next*, *Finish* and other available buttons, do the following for each button:
        1. Select the button.
        2. In the *Properties Explorer*, under *Behavior — Enabled*, select *Advanced*.
        3. In the *Dependent Property Editor*, select *Calculation Rule*.
        4. Write a calculation rule to control the `Enabled` and `Visible` properties of the button.
4. **Configure the steps.**
    1. Configure the *Review* step:
        - Display an overview of the steps performed, with read only controls and transparent tables.
        - Disable the *Next* button.
        - Enable the *Finish* button and configure it such that on click it shows the *Confirmation* step.
    2. Configure the *Confirmation* step:
        - Hide the *Cancel* and *Finish* buttons.
        - Show the *Close* button.
        - Add a section group to the form region; with the header, **`What do you want to do next?`**
        - Add *Links* to the section group and configure the navigation.
          For example, the links could be to `Create another business object instance` or `Edit business object instance`.
    3. Configure the floorplan title, action buttons and create an OBN Inport similar to a quick activity configuration.
       For more information, see Model a Quick Activity Floorplan [page 700].
5. **Save and activate your changes.**

9.3.2.7 Model a Work Center View

Prerequisites

- The required OWL has been created.
- If you want to create a list of common tasks, then the corresponding target floor plans, which implement the OBN operations used in this view, have to be declared as assigned objects.
- If there are common tasks and related links that cannot be specifically assigned to one view and belong to every view of a work center, an Include View with these links is created.

Procedure

1. **Create a work center view**.
    1. In the *Solution Explorer*, right-click your project and select *Add* > *New Item*.
    The *Add New Item* dialog box opens.
    2. Select *Work Center View*.
    3. In the *Name* field, enter a name for the work center view and click *Add*.
    The work center view is added to your project.
    4. Double-click the work center view.
    The work center view opens in the user interface designer.
    5. Check the following properties of the work center view in the *Properties Explorer*:

Property	Description
Configuration — Help ID	ID of the related help resource
Configuration — Support Area Code	Used in ITSAM, for example to propose a CSN component for an incident
Configuration — View SubType	Type of the view, for example, `Overview`, `Application`, `IncludeView`
Extensibility — Anchor	Associated flexibility anchors
Misc — Name	Name of the User Interface Component
RBAM Data — Authorization Classification Code	Classifies the RBAM start authorization behavior. For normal views, the entry should always be `ApplicationContainer`.

2. **Embed an OWL**.
    1. Drag and drop the OWL from the *Configuration Explorer* to the *Designer* tab.
    2. Click *Bind*.
    3. In the *Embedded component binding* dialog box, bind the OWL to the work center view.
    For more information, see Model an Embedded Component [page 716].

3. **Define the assigned objects**.
   All the assigned objects together define the complete authorization granted to a user assigned to the view. Assigned object also specify the collection of valid navigation targets for a view and the elements that should appear in the *Reports* view of a work center.
   1. In the *Properties Explorer*, select *WorkCenterView*.
   2. Under *RBAM Data — Assigned Objects*, click *...*.
   3. In the *Add Assigned Objects* dialog box, click *Add* and then click *...* to select the required Component Id.
   4. Select the required *Assigned Object Type*:
      - *Navigation Target*: To specify the UI components that should be reachable from this view.
      - *WocViewInclude*: To include views of type Include View.
      - *TaskList View*: A collection of BTM task types that should be supported by the view.
      - *Reports view*: To allow the view to show a list of reports in the work center.

        > **i** Note
        >
        > The reports are assigned in the *Business Analytics* work center.
   5. If you want to define the dependency of the assigned objects, define the scope rules.
   6. If there are a set of navigation targets that belong to many views of a work center, create an *Include View* and assign the *Include View* as an assigned object to this view.
      An *Include View* is built like any other view with a few exceptions like:
      1. You cannot include Embedded Components.
      2. The View sub-type has to be *Include View*.
      3. An *Include View* cannot have another Include View.
      4. Access context cannot be maintained for Include View.
      5. An assigned object of type Include View does not have scope rules.
   7. Click *OK*.
4. **Specify the common tasks and related links specific to this view.**
   1. In the *Properties Explorer*, select *WorkCenterView*.
   2. Under *Links — CommonTasks*, click *...*.
   3. In the *NavigationListItem Collection Editor*, click *Add*.
   4. Under *Data — Link*, click *...* .
   5. In the *Model Navigation ...* dialog box, select an existing OBN navigation or create a simple navigation. For example, you can select an existing OBN navigation for the common task Create a Sales Order.
   6. You can also create the required event handlers for the common tasks.
      For more information, see Create an Event Handler [page 747].
   7. If you want a list of related links in the work center view, repeat the above steps under *Links — Related Links*.
5. **Define the enterprise search objects.**
   1. In the *Properties Explorer*, select *WorkCenterView*.
   2. Under *RBAM Data — Enterprise Search Objects*, click *...*.
   3. In the *Enterprise Search Objects Editor*, assign the required enterprise search controller objects to the view and click *OK*.
      The users assigned to this view are authorized to use Enterprise Search and the assigned objects are available in the drop down list of available enterprise searches.
6. **Define the external applications.**
   1. Under *RBAM Data — External Application Data*, click *...* .

2. In the *Select External Application Data* dialog box, assign the external applications and click *OK*.
   The service interfaces in the assigned external applications can now be used by this view.

7. **Specify the access context.**
   If the view supports instance restrictions, you can maintain access contexts.
   1. Under *RBAM Data — Access Context Code*, select the required access code of the view.
8. **Save and activate your changes.**

9.3.2.8 Model a Work Center

Overview

You can model a work center floorplan to display the various work center views.

Prerequisites

You have created the required work center views.

Procedure

1. **Create a work center with default views**.
   1. In the *Solution Explorer*, right-click your project and select *Add* > *New Item*.
      The *Add New Item* dialog box opens.
   2. Select *Work Center*.
   3. In the *Name* field, enter a name for the work center and click *Add*.
      The work center is added to your project.
   4. Double-click the work center.
      The work center opens in the user interface designer.
   5. Check the following properties of the work center in the *Properties Explorer*:

Property	Description
Configuration-Help ID	ID of the related Wektra help resource
Extensibility-Anchor	Associated flexibility anchors
Misc- Name	Name of the User Interface Component

RBAM Data-Authorization Classification Code	This indicates how the component is handled for startup authorization. If there is an embedded work center view the entry should always be **Application Container**.

2. **Assign work center views to the work center**.
    1. On the *Designer* tab, click on the folder icon.
    2. In the *Properties Explorer*, select *OberonCenterStructure*.
    3. Under *Misc-ViewSwitches*, click *...* .
    4. In the *View Switch Collection Editor*, add a new view switch for each of the work center views as follows:
        1. Click *Add*.
        2. Click on the work center views you have added and under *Misc- Name*, enter a name for the view switch.
        3. Click *OK*.
    5. To add a sub view:
        1. In the *View Switch Collection Editor*, select the view switch.
        2. Under *Misc-SubViewSwitches*, click *...* .
        3. In the *Sub ViewSwitch Collection Editor*, click *Add*.
        4. Click on the sub views you have added and under *Misc-Name*, enter a name for the sub view switch.
        5. Repeat the steps for the required number of sub views.
    6. To assign a work center view to a view switch:
        1. In the *Designer* tab, select the view switch.
        2. From the *Configuration Explorer* drag and drop the work center view, to the *Designer* tab.
        3. Repeat the steps for all the view switches and sub view switches.
3. **Configure the work center**.
    1. Specify the scope rules for the work center and the view switches:
        1. In the *Properties Explorer*, select the work center.
        2. Under *Configuration-ScopeRules*, click *...* .
        3. In the *Scope Rules Editor*, assign the required scope rules and click *OK*.
        4. Repeat the steps for all the view switches.
4. **Save and activate your changes**.

9.3.2.9 Create a Port Type Package

Overview

You can create a port type package in the studio and modify its attributes in the user interface designer.

A port type package is a component that hosts one or more port types. A port type is a blueprint for an inport or an outport. Navigation between components that have inports and outports within the same port type package can be facilitated without any mapping effort.

Prerequisites

You have created a solution in the studio.

Procedure

1. In the *Solution Explorer*, right-click your project and select *Add* > *New Item*.
2. The *Add New Item* dialog box opens.
3. Select *Port Type Package*.
4. In the *Name* field, enter a name for the port type package.
5. Click *Add*.
   The port type package is added to your project.
6. Double-click the port type package.
   The port type package opens in the user interface designer.
7. Create the required structure, as follows:
   - To add a port type, select the root node and then click +.
   - To add a parameter, right click on the port type and choose *Add Parameter*.
   - To add a List Parameter, right click on the port type and choose *Add List Parameter*.
     You can add parameters under a list parameter.
   - To set a parameter as a key parameter, right click on the parameter and choose *Set as Key*.
8. To rename any entity in the port type package, right click on the entity and choose *Rename*. Type the required name and press `Enter`.
9. Save and activate your changes.

9.3.2.10 Model a Modal Dialog

Overview

You can model a modal dialog to create a dialog box that is triggered by an event action. You can model a reusable modal dialog and then bind it to a floorplan, or you can model a modal dialog that is specific to a floorplan and that can be called by an event handler.

Prerequisites

You have created a solution in the studio.

A. Model a reusable modal dialog.

Procedure

1. In the *Solution Explorer*, right-click your solution and select *Add* > *New Item*.
   The *Add New Item* dialog box opens.
2. Select *Modal Dialog Floorplan* and click *Add*.
   The modal dialog is added to your project.
3. Double-click the modal dialog.
   The modal dialog opens in the user interface designer.
4. Select the identification region.
5. In the *Properties Explorer*, under *Misc – Window Title*, click the arrow button.
6. In the *Dependent Property Editor*, enter the title and click *OK*.
7. On the *Designer* tab, drag the required controls from the *Toolbox* to the panes.
8. On the *Data Model* tab, bind each of the fields to the required business object element.
   For more information, see Bind Data Elements to Business Object Elements [page 731].
9. Create the required event handlers.
   For more information, see Create an Event Handler [page 747].
10. Create the bindings in the modal dialog as follows:
    1. On the *Controller* tab, right-click *Bindings* and add a binding.
    2. Select a field or a structure for the binding.
    3. Enter a *Binding Name*.
    4. Click *Bind*.
11. In the parent component, do the following:
    1. On the *Controller* tab, right click *Modal Dialog* and add a modal dialog.
    2. In the *Repository* dialog box, navigate to the modal dialog that you have created.
    3. Click *OK*.
       The modal dialog is added to the parent component.
12. On the *Controller* tab of the parent component, click the new modal dialog and then under *Binding Configuration* do the following:
    1. In the *Parent data model*, select the field for the binding.
    2. In the *Controller Interface for Embedded Component*, select the binding.
    3. Click *Bind*.
       The modal dialog is now bound to the parent component. This facilitates the synchronous passage of data from the parent component to the modal dialog.
13. Save and activate your changes.

B. Model a modal dialog specific to a floorplan

Procedure

1. In the *Solution Explorer*, double-click the floorplan where you want to add a new modal dialog.
   The floorplan opens in the user interface designer.
2. From the *Toolbox – Panes*, drag and drop a *ModalDialog* onto the *Designer* tab.
3. In the new modal dialog, do the following:
    1. In the *Properties Explorer*, under *Text Information – Title*, click the arrow button.

2. In the *Dependent Property Editor* dialog box enter the required title in the *Overridden text* field.
3. Click *OK*.
4. On the *Designer* tab, drag the required controls from the *Toolbox* to the panes.
5. On the *Data Model* tab, bind each of the fields to the required business object element.
   For more information, see Bind Data Elements to Business Object Elements [page 731].

4. On the *Controller* tab, right-click *Event Handler* and add a new event handler.
5. For that event handler do the following:
   1. Select operation type as *Show Modal Dialog*.
   2. Under *Configure the Operation-Modal Dialog name* select the modal dialog you have created.
6. Assign the event handler to a button or link.
   For more information, see Create an Event Handler [page 747].
7. Save and activate your changes.

9.3.2.11 Model an Embedded Component

Embedded components are reusable sub screens that facilitate:

- Inport and outport navigation between the parent component and the embedded component.
- Bindings to reference data from the parent component.
- Configuration to let the parent component influence the behavior of an embedded component.

An embedded component can be integrated with its parent component as follows:

- *Tight coupling*: This is handled by referencing a parent data model either explicitly by creating bindings in the embedded component interface, or implicitly by node reference binding. For example, tight coupling is used in custom panes, address maintenance, and business cards.
- *Loose coupling:* Embedded components are triggered in an inport context mapping, for example, by passing an ID. An embedded component can pass back parameters to a parent component using outport context mapping. For example, mashups and information snippets displayed in a fact sheet. If an embedded component is loosely coupled, it should be self-contained and the data binding and controller logic needs to be defined in the embedded component.

You can delete the embedded component when your solution is in maintenance mode. For more information, see Maintenance Mode [page 185].

Procedure

1. **Create an Embedded Component**.
   1. In the *Solution Explorer*, right-click your project and select *Add* > *New Item*.
      The *Add New Item* dialog box opens.
   2. Select *Embedded Component*.
   3. In the *Name* field, enter a name for the embedded component and click *Add*.
      The embedded component is added to your project.
   4. Double-click the embedded component.
      The embedded component opens in the user interface designer.

You can model an embedded component just like any other UI component. The main difference is in the controller interface binding, controller configuration and node reference binding.

2. **Prepare to tightly couple an embedded component with its parent component.**
    1. To tightly couple an embedded component by explicit binding, in the embedded component do the following:
        1. In the *Controller* tab, right-click on *Bindings* and choose *Add Binding*.
        2. Select the newly created binding.
        3. Select a field or a structure for the binding.
        4. Click *Bind*.
        5. In the *Binding* field, enter a name.
        6. Save and activate your changes.
    2. To tightly couple an embedded component by node reference binding, in the embedded component do the following:
        1. Drag the required fields to the *Designer* tab.
        2. On the *DataModel* tab, select an embedded component node, for example, Root.
        3. In the *Properties Explorer* under *Node Reference Binding*, set *Virtual Binding* to `true`.
        4. On the *Controller* tab, right-click on *Bindings* and choose *Add NodeReference Binding*.
        5. Select *NodeReference* and select the node on which the Virtual Binding property was set to true.
        6. Click *Bind*.
        7. Save and activate your changes.
3. **Bind the embedded component**.
    1. Explicit binding
        1. Open the parent component.
        2. In the *Configuration Explorer*, navigate to the embedded component you created.
        3. Drag and drop the embedded component to the *Designer* tab.
        4. Click *Bind*.
        5. In the *Embedded component binding* dialog, in the *Bindings Configuration* tab, select an attribute or structure from the Parent Data Model and select a binding under the Controller Interface of Embedded Component.
    2. Node reference binding
        1. Open the parent component.
        2. In the parent component, create a data structure.
        3. In the *Configuration Explorer*, navigate to the embedded component you created.
        4. Drag and drop the embedded component to the *Designer* tab.
        5. Click *Bind*.
        6. Bind the data structure created in the parent component to the node reference binding created in the embedded component.
4. **Configure the embedded component.**
   An embedded component can expose configurations, to let the parent component influence the behavior of the embedded component, through configuration parameters and configuration list parameters.
    1. Add the required configuration parameters:
        1. In the *Controller* tab, under *Configuration*, right-click *Parameters* select *Add ConfigParam*
        2. Choose either *Value* or *Value Help*.
           Data is passed from the parent component to the embedded component. *Value* is the default value to be used by the embedded component in case it is not specified by the parent component. *Value*

Help is used when the options to be selected by the parent component are known and fixed. A default value can be selected in *Value Help* too.
For example, if the configuration parameter is LayoutType, then the options such as TableLayout, NewsPaper Layout could be added as value help and the table layout could be set as default. In the Parent component you can change it to NewsPaper Layout.

> **i Note**
>
> You can group a few configuration parameters in a configuration parameter structure.

3. Create a configuration parameter list under *Configuration*, by right-clicking on *Parameter List*. The *MinOccurs* field property has a fixed value of 1 in the case of a *Configuration Parameter List* .
4. Specify the minimum and maximum amount of entries allowed in the list, by setting the *minOccursField* and *maxOccursField* properties of the configuration parameter list.

> **i Note**
>
> A data model entity is created for each configuration parameter, structure and parameter list. You can configure their properties in the *Properties Explorer*.

5. **Configure the parent component.**
    1. To use the configuration parameters of the embedded component, open the parent component.
    2. Select the embedded component.
    3. In the *Properties Explorer*, under *Parameters — StructureParameters*, click *...* .
    4. In the *StructureParameter Collection Editor*, to add values for structure parameters do the following:
        1. Click *Add*.
        2. Select the added structure parameter and under *Misc-Name* enter the name.
        3. Under *Parameters-Parameter* click the arrow button.
        4. In the *ECParameterValueEditor* enter or choose the parameter value and click *OK*.
        5. In the *StructureParameter Collection Editor* click *OK*.
    5. Save and activate your changes.

Alternatives

You can also loosely couple an embedded component to a parent component.

1. Create an embedded component.
2. In the *Controller* tab, model the inports, in the followings manner:
    1. Right-click on *Inports* and add a new inport.
    2. Add a new parameter and bind this to a data field in the data structure.
3. Model the outports to pass back values in a similar manner as the inports.
4. Open the parent component.
5. In the *Controller* tab, model the outports, to access the embedded component.
6. Model the inports to pass back values.
7. In the *Configuration Explorer*, drag the embedded component to the *Designer* tab and click *Bind*.
8. In the *Embedded component binding* dialog, in the *Simple Navigation* tab, right-click on *Navigation* and select *Add Navigation*.

9. Map the parent component outport with the embedded component inport and bind the parameters.
10. Right-click on *Navigation* and select *Add Navigation*.
11. Check *Reverse Navigation*.
12. Map the embedded component outport with the parent component inport and bind the parameters.
13. Click *OK*.
14. Save and activate your changes.

Result

You should note that once an embedded component is used in a parent component, any interface changes may be incompatible.

9.4 Modify Your Floorplans

9.4.1 Floorplan Modification Quick Guide

This document gives you an idea about the various modifications and enhancements you can make to the floorplans created by you using the user interface designer.

For more information about creating floorplans, see the Floorplan Modeling Quick Guide [page 694].

Business and technical background

SAP's User Interface Style Guide

You can learn how to build user interfaces according to this style guide.

Transformations

A business object (BO) is modeled generically, and not for a specific user. The BO structure and node attributes, therefore do not always fit the needs of a specific user interface.

A transformation allows the conversion of a BO node attribute into a form that can be used on the user interface.

For more information, see here [page 724].

Tasks

Fields

Bind Data Elements to Business Object Elements

You can bind the data elements you have added to your floorplans, with the business object elements in the backend, so that data can be derived from the backend when a particular data element is used at runtime.

For more information, see here [page 731].

Define Dynamic Behavior of a Field

A dynamic expression is used to define the behavior of a field within a specific context. You can for example, define the visibility of a field or a group of fields depending on the content of another field on the screen.

For more information, see here [page 732].

Buttons

Configure a Button

You can add any required button on the application toolbar. The *Cancel* and *Close* buttons are configured in the system. If you add these buttons, the system automatically associates the event handler, to close the floorplan. If you want to any other button, you need to write the required event handler. You can also configure a button to open a floorplan by configuring an object based navigation between floorplans.

For more information, see Configure a Button [page 733].

Create a Preview Button for a Print Form

You can create a *Preview* button and link it to a print form. Clicking the *Preview* button allows you to preview the form as a PDF file and provides options to either save or print the form.

For more information, see Create a Preview Button for a Print Form [page 734].

Translate a Mandatory Button

When you create a solution in a language other than English, you can translate the labels of mandatory buttons in the UI designer.

For more information, see .

Screen Sections

Create a Pane in a Floorplan

You can create a pane to display different types of controls on a screen.

For more information, see here [page 737].

Create a Preview Pane

You can create a preview pane to preview information about a business object listed in a table.

For more information, see here [page 738].

Screen Layout and User Interface Texts

Change the Title of a Component

You can change the text of any component visible on the user interface.

For more information, see here [page 739].

Hide a User Interface Component

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Select the user interface component you want to hide, for example, a section group, or a field.
3. In the *Properties Explorer*, under *Behavior — Visible*, set the value to `False`.
4. Save and activate your changes.

Create a Text in the Text Pool

For more information, see here [page 740].

Object Worklists (OWLs)

Configure a Query

You can create and configure a query to view specific instances of a business object in a floorplan.

For more information, see here [page 741].

Tables

Reorder Columns in a List or a Table

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. In the *Properties Explorer*, under *Child Elements – List Columns*, click *...* .
3. In the *Column Collection Editor*, select the column you want to move.
4. Use the up and down arrows to move the column to the required position.
5. Save and activate your changes.

Navigation between floorplans

Configure an Object-Based Navigation in a Target Floorplan

You can use object-based navigation (OBN) to navigate between floorplans and pass parameters between floorplans.

For more information, see here [page 742].

Configure an Object-Based Navigation in a Source Floorplan

You can use object-based navigation (OBN) to navigate between floorplans and pass parameters between floorplans.

For more information, see here [page 744].

External Navigation

Add a Link to Trigger an External URL

You can add a link that opens an external URL in your floorplans.

For more information, see here [page 745].

Transformations

Configure a Dedicated Field

You can add a dedicated field to a data model and use a field transformation to calculate the value of this dedicated field from existing business object (BO) node attributes.

For more information, see here [page 750].

Configure a Default Set Parameter using Transformations

You can use a transformation to calculate the value of a default set parameter .

For more information, see here [page 751].

Additional Tasks

Write a Script

You can write front-end scripts while creating an event handler and while defining a calculation rule.

For more information, see here [page 335].

Create an Event Handler

You can create an event handler to respond to user actions on the screen.

For more information, see here [page 747].

Configure an Action Form

You can create action forms to use the action parameters of a business object (BO) node or an enhanced controller object (ECO) node. Action forms are always bound to a BO/ECO node's ESI action that has parameters. The binding is done within the BO Action operation of an event handler. Action forms belong to a user interface component and cannot be reused in other user interface components.

For more information, see here [page 748].

Configure an Object Value Selector

You can configure an Object Value Selector (OVS) control to open an OVS dialog box.

For more information, see here [page 749].

Add a Report List View to a Work Center

You need to assign standard report list views, for example the List and Gallery standard reports list views, to a work center in the user interface designer.

For more information, see here [page 752].

Define an Anchor for a Floorplan

You can define anchors for your floorplans to allow end users to make changes to the floorplans provided by you.

For more information, see here [page 753].

Save and Activate Your Changes

After you have created any of the changes listed above, you have to save and activate the changes.

1. From the *File* menu, choose *Save*.
2. From the *File* menu, choose *Activate*.
3. In the *Activate Worklist* dialog box, select the check boxes beside the relevant components and click *OK*.

9.4.2 Business and Technical Background

9.4.2.1 Transformations

Overview

A business object (BO) is modeled generically, and not for a specific user. The BO structure and node attributes, therefore do not always fit the needs of a specific user interface.

For example, in the Purchasing contracts BO , a specific value needs to be entered to ensure the unlimited validity of contracts. So a date like 31/12/9999 is used for this purpose. On the user interface however, a checkbox to indicate unlimited validity is needed and not a set date.

A transformation allows the conversion of a BO node attribute into a form that can be used on the user interface.

You can use transformation rules to calculate the value of fields or parameters in the user interface data model that do not have direct counterparts within the BO node structure.

Examples of **transformation rules** include:

- Period-to-Indicator

- Code-to-Indicator
- Concatenation of attribute values
- Time-to-Duration
- Age from date

A transformation has one or many inbound parameters. The input for a transformation can either be a constant value, a binding against a field of a BO node structure, or an already existing data field from the user interface. The resulting outbound value is assigned to a target field or a parameter inside the user interface data model.

The type of transformation rule used depends on whether the value being generated is for use in a dedicated field or in a default set parameter.

For each namespace, a predefined set of transformation rules are available.

Related Information

Configure a Dedicated Field [page 750]
Configure a Default Set Parameter using Transformations [page 751]

9.4.2.2 Operation Types

Overview

Operation types in the user interface designer allow you to assign an activity to an event handler.

When creating an event handler [page 747], you can select from one or more operation types.

Following are the various operation types:

BOAction	ContainerExit	FireEventHandler	OpenLink	WindowAction
BOOperation	DataOperation	FireOutport	ShowModalDialog	Controller
CheckAndDetermine	DocumentOutput	List	SynchDataContainer	RestartClient
Condition	ExecuteDefaultSet	MessageBox	ValueHelp	QCAfterSaveAction

BOAction

The *BOAction* can be used to invoke a special function exposed by an ESR action.

BOAction parameters can be bound to data fields in the data model to pass values to parameters. To enter values for the action parameters, you can configure an action form in the BOAction. Add the controls to the

action form and bind the action parameters to the data fields bound to controls. On executing the event handler, the action form is displayed. Enter the action parameter values and click OK to execute the BOAction.

You can configure the following additional parameters for the BOAction:

- Instance Binding is the binding of an action against the node instance(s). It is required for list-binding.
- Save Before Execution indicates whether or not a save before the execution of the action is required, for example for one-click actions.
- Save After Execution indicates whether or not an immediate save is required.
- Continue on Reject allows the execution to continue even when the BOAction fails. If this flag is not checked, the event handler is stopped.
- Special Reference Action supports copy and paste for lists. The list operation CutCopy stores the selected instances in the buffer. To use this selection for the next reference action, pasting in this case, you can model a special reference BOAction by checking this flag. Then, the selected instances in the first operation, CutCopy in this case, are passed as parameters for this second reference action.

BOOperation

The *BOOperation* is an implicit ESI-Action. This operation is supported by all business objects even though it is not explicitly modeled in ESR as an action. It is used to perform Read, Create, DelayedCreate and ValueHelp.

You can configure the following additional parameters for the BOOperation:

- Set Suppress Messages to true to suppress all messages invoked by triggering this operation in the backend. Otherwise, all ESI and backend messages will be displayed.
- Select Namespace and Select BO describe the BO on which the operation is performed.
- BO Operation parameters are defined by the user and not derived from ESR metadata.
  For example, for a read operation, you need to pass the ID value to indicate with which ID you want to invoke the read.
  There are the following parameter types:
  - the keyNavigation parameter is passed with special metadata, here the key flag and GDT metadata.
  - With nodeID, the value of the node id is passed
  - The alternativeKey is modeled for some BOs as number ranges, for example. In the backend, they need special treatment before a read/retrieve can be done.
    The value of the alternative key node is converted to node ID. The type of alternative key is indicated by the data field bound to the path attribute. This value is used at runtime to identify the type.

CheckAndDetermine

The *CheckAndDetermine* offers a generic function for checking BOs, without the need for an explicit, modeled BO action. A check carried out with this operation triggers the validation of a whole BO instance, or a specific node with subnode tree. Error or warning messages are sent to the client if there are any inconsistencies.

This operation is generally used for Guided Activity floorplans, but may also be used in other applications where you want to implement a check button.

Condition

Condition is used as a branch control to call different operations based on a data model value or constant. You can specify a default option, as well as configure multiple select options with different compare types and values; and nested condition operations.

ContainerExit

ContainerExit is an action defined on the exitClass and should only be used in special situations, namely, wherever a data structure has an exit class assigned.

You can configure any one of the actions in the action field, or create new ones manually. If you use an existing action, the system automatically sets the exit class property of the structure you have selected as path.

DataOperation

DataOperation allows you to modify the value of data model fields.

- Assign the value of a text pool entry to the target data field with assignText.
- Assign the value of the source data field to the target data field with assign.
- Assign a constant value to the target data field with assignConstant.
- Clear the value of the target data field with clear.

DocumentOutput

DocumentOutput configures Microsoft Excel and Microsoft Word downloads. The group ID is used by the Excel/ Word export to identify the assigned template group.

ExecuteDefaultSet

ExecuteDefaultSet executes an ESI query. Set Blocking to block access to the UI during query execution.

FireEventHandler

FireEventHandler allows you to invoke other event handlers.

FireOutport

FireOutport is used to fire and pass data to the outport. Set TakeOverMessagesToNewWindow to carry forward messages from the previous save to a new window in Save and New scenarios.

List

List triggers operations related to the data list such as Add Row and Remove Row. Set Blocking to block access to the UI during the operation.

MessageBox

MessageBox configures decision pop-ups, for example `Are you sure you want to delete?` A deletion template is available, as well as other parameters to customize your configuration.

Use List Binding to enable plural text and list count for multi-selection.

OpenLink

OpenLink opens a link in an external browser. The link value can be bound to a data field and URL.

Set the browser target features as follows:

- Default = empty
- Status = yes
- Toolbar = yes
- Menubar = yes
- Resizable = yes

Set Process Protocol to explicitly switch off the protocol handling for the OpenLink operation. In this way, you ensure that file:// or https:// remains in the URL.

Script allows you to define front-end script for dynamic expressions. Set Blocking to block access to the UI during script execution.

In the Scope Fields section, you can configure data fields that trigger script execution. When a field value changes, the script is executed.

ShowModalDialog

ShowModalDialog allows you to display an in-component modal dialog as pop-up To configure it, drag and drop Modal Dialog from the toolbox to the user interface designer or create one from the selection proposal in this operation.

SynchDataContainer

SynchDataContainer forces a backend round trip and flushs the data container at runtime. You can use this operation instead of, for example, provoking a round trip by flagging a dummy field with requiresRoundTrip and setting a value in this field.

When the Finalize flag is set, DoPostProcessing is invoked which results in the evaluation and sending of messages.

ValueHelp

ValueHelp should be configured if an object value selector (OVS) component is involved.

- Set the Use Query flag to trigger the query configured as query attribute.
- Query Name is the query in the OVS to be called.
- For Operation Type, **Get** is for retrieving DefaultSets, and **Set** for passing to the parent component.
- Use Suppress Messages to suppress messages coming in the value help round trip.
- Parent Component Id and Parent Path are the parent component node information for the RetrieveValueSet call.
- Select Namespace and Select BO Model describe the OVS controller name.

WindowAction

In *WindowAction*, set TakeOverMessageInCallingWindow to Yes to show messages of the current operation in the calling window.

For Success Notification Port, configure the outport to be fired on save. The outport and inport at the source should be configured with the same local event name. In this way, when a quick activity is opened from a work list, then saved and closed, the system informs the port that the save is successful and the work list is refreshed.

Set Continue on Reject to continue with the execution even if the window action fails.

Controller

Controller should be used to create Actions on BO in case of new Controller Infrastructure. From a functional perspective a controller acts like a BO Action in a new controller infrastructure.

RestartClient

RestartClient clears details from the DataModel and reloads the screen from the backend. In this way, if you open a screen, make a change, then cancel the operation, the screen reverts to its state before the edit operation was performed. There is no configuration for this operation.

QCAfterSaveAction

QCAfterSaveAction is configured in the Quick Create (QC) component after a *WindowAction* operation. You can configure the action type to open the Thing Inspector after the save operation, or open a new Quick Create screen.

9.4.2.3 Message Mapping

Overview

A message is the text shown at the bottom of the user interface (UI) during runtime. It presents the user with a piece of information, an instruction, or a warning in a given situation. In SAP cloud solutions messages are sent to the UI from the business object (BO) that the UI is based on. The messages are part of the infrastructure delivered by SAP or can be part of a custom BO.

In some cases, the original messages of the BO that are shown during runtime do not fit the context of the UI. For example, they have the wrong terminology or the terms used are not specific enough. In those cases, you can exchange messages or change the behavior of messages on the UI by using the message mapping offered by the UI designer. If the original message contains parameters you can also display a different message for each individual parameter value.

You map messages only within the context of a specified UI and not across the system. This means, if a message is shown in the context of two different UI models, you can map it for one UI model and this mapping does not affect the message in any other UI model.

Prerequisites

Message mapping in the UI designer is only available for BOs that contain messages.

Structure

In the UI designer, message mapping is located on the *Controller* tab of those UI components that directly communicate with the BO or the backend. On the *Controller* tab, the following sections deal with the messages of a BO:

- Messages
  In this section, the UI designer displays all messages associated with the BO that the currently selected UI component is based on.
- Message Mappings
  In this section, you can see the original messages associated with the BO and you can do the actual mapping. For more information, see Map a Message [page 754].
- Message Mapping Overview
  In this section, you see the original messages together with the mapped messages.

In the UI designer, you can use the following rules to deal with messages:

- Replace
  The original message is replaced by a new message. During runtime, the system displays the new message on the UI instead of the original message.
- Ignore
  The system ignores the original message and does not show it during runtime on the UI.
- Original
  The system displays the original message during runtime.
- Aggregate
  Any number of messages can be combined into one new message. The system displays this new message on the UI during runtime instead of the original messages.

9.4.3 Tasks

9.4.3.1 Bind Data Elements to Business Object Elements

Overview

You can bind the data elements you have added to your floorplans, with the business object (BO) elements in the backend, so that data can be derived from the backend when a particular data element is used at runtime.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. In the *Data Model* tab, under the Root structure, add the necessary data fields, data structures, and data lists.

3. Bind the data structures and the data fields under the Root structure to the BO structure nodes and the BO elements.
   To bind a data element, select the required data element from the left list and select the required business object element from the right list and click *Bind*.
4. In the dialog box that opens, click *Yes* if you want the system to automatically use the text maintained for the BO element as the field label.
5. If you want to manually enter a field label in the user interface designer, in the *Designer* tab do the following:
   1. Double-click the user interface control.
      The *Dependent Property Editor* dialog box opens.
   2. Enter the field label in the *Overridden Text* field.
6. Save and activate your changes.

9.4.3.2 Define Dynamic Behavior of a Field

Overview

A dynamic expression is used to define the behavior of a field within a specific context. You can for example, define the visibility of a field or a group of fields depending on the content of another field on the screen.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Select the required field.
3. In the *Properties Explorer*, under *Behavior – Visible*, select **`Advanced`**.
4. In the *Dependent Property Editor*, select the variant *Calculation Rule*.
5. Under *Calculation Rule*, define the formula as a Boolean expression.
   For example, if the formula is the following:

```
($Data.DiscountType == 2 ),
return true
else
return false
```

   Then, this field is displayed only when the *Discount Type* field has the value 2.

   > **i** Note
   >
   > To define the formula, you can use every field that is defined in the *DataModel*. The fields available in the *DataModel* are listed under the collection pattern *Invalidation Trigger*.

   For more information, see Write a Script [page 746].
6. Under *Invalidation Trigger*, mark each of the fields you have used within the formula and add it using the arrow buttons.

7. Click *OK*.
8. Save and activate your changes.

Example

There are many discount types available in the system. If the user selects the **Discount Type** `Christmas`, you want the **Minimum Bill Amount** field to be visible on the screen. This field is otherwise not visible if the user selects any other discount type. In such a case, you can define a dynamic expression for the **Minimum Bill Amount** field.

9.4.3.3 Configure a Button

Overview

You can add any required button to the application toolbar of your floorplan. The *Cancel* and *Close* buttons are configured in the system. If you add these buttons, the system automatically associates the event handler to close the floorplan. If you want to add any other button, you need to create the required event handler. You can also configure a button to open a floorplan by configuring an object-based navigation between floorplans.

Prerequisites

You have created a floorplan.

Procedure

1. In the studio, double-click the floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Open the *Configure Toolbar* as follows:
   1. In the *Properties Explorer*, select the *AdvancedListPaneVariant* to which you want to add a button.
   2. Under *Toolbar Information - Use Toolbar*, select **`True`**.
   3. Under *Toolbar Information - ListPaneToolbar*, click ... .
3. Add the required button in the *Configure Toolbar* dialog as follows:

Task	Action

To add a configured button	Under *Available* list, select *Close* or *Cancel* button and click the right arrow button. The button is now available in the *Current* list.
To add an application-specific button and configure it manually	1. Under *Available* list, select the required button or the *Application Specific* button and then click the right arrow button. 2. Under *Events - OnClick*, select *New Event Handler*. For more information, see Create an Event Handler [page 747].
To add a button and configure an object-based navigation (OBN)	1. Under *Available* list, select the required button or the *Application Specific* button, and then click the right arrow button. 2. Configure OBN navigation in the source and target floorplans. For more information, see Configure an Object-Based Navigation in a Source Floorplan [page 744] and Configure an Object-Based Navigation in a Target Floorplan [page 742].

4. Click *OK*.
5. Save and activate your changes.

9.4.3.4 Create a Preview Button for a Print Form

You can create a *Preview* button and link it to a print form. Clicking the *Preview* button allows you to preview the form as a PDF file and provides options to either save or print the form.

Prerequisites

- An activated floorplan exists for the business object.
  For more information, see Model an Object Instance Floorplan [page 702], steps 1–3.
- An activated print form is associated with the business object.
  For more information, see Create a Print Form [page 508].

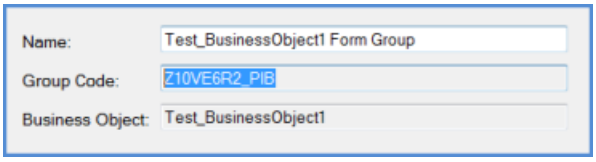
Procedure

1. **Open the floorplan in the user interface designer (UI designer) and create an outport.**
   1. In the studio, double-click the floorplan in the *Solution Explorer*.
      The floorplan opens in the UI designer.
   2. On the *DataModel* tab, right-click on Root and select *Add Data Field*.

3. In the *Properties Explorer*, set the following properties of the data field:
   - *Initial Value*: **`True`**
   - *Type*: **`Boolean`**
   - *Name*: **`bool_true`**
4. On the *Controller* tab, right-click on *Outports* and then select *Add Outport*.
5. To rename the outport, right-click the outport and select *Rename*. Enter a name, for example, **`PreviewOutport`**.

2. **Add the outport parameters.**
   1. In the *Outport Configuration: OutPort* view, click the *Add Parameter* button.
   2. In the *ParameterName* field, enter **`GroupCode`**.
   3. In the *Constant* field, enter the group code.

      > → Tip
      >
      > You can find the group code in the studio. In the Solution Explorer, double-click the .ftgd file. The group code is displayed as shown in the figure below.

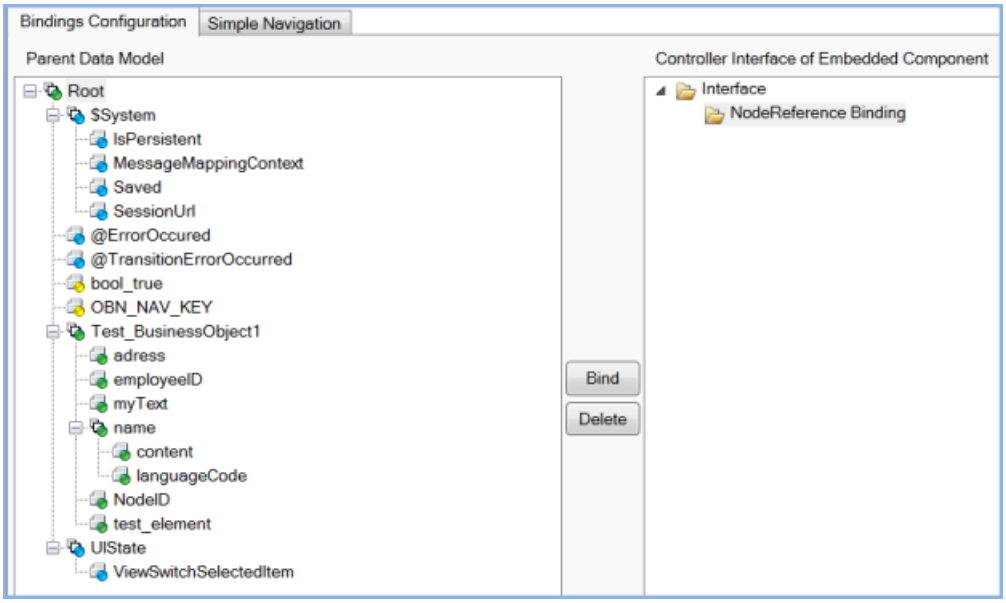
      

      Group code displayed in the studio

   4. Click the *Add Parameter* button again.
   5. In the *ParameterName* field, enter **`ForceFrontendPreview`**.
   6. Next to *ParameterBinding*, click .... The *Parameter Binding* dialog box opens.
   7. Select the data field **`bool_true`** and *OK*, then save your changes.
      The output parameters have been entered as shown in the figure below.

      Outport parameters entered in the Parameters Binding dialog box

3. **Bind the outport and inport parameters.**
   1. On the *Controller* tab, right-click on *Modal Dialogs* and then select *Add ModalDialog*.
   2. In the *Repository Browser* dialog box, navigate to the modal dialog named `Preview` by selecting **`SAP_BYD_APPLICATION_UI –> Reuse –> OutputManagement –> Preview`** and click *OK*.
   3. Under *Modal Dialogs*, click *Preview*.
      The *Bindings Configuration/Simple Navigation* dialog box opens as shown in the following figure.

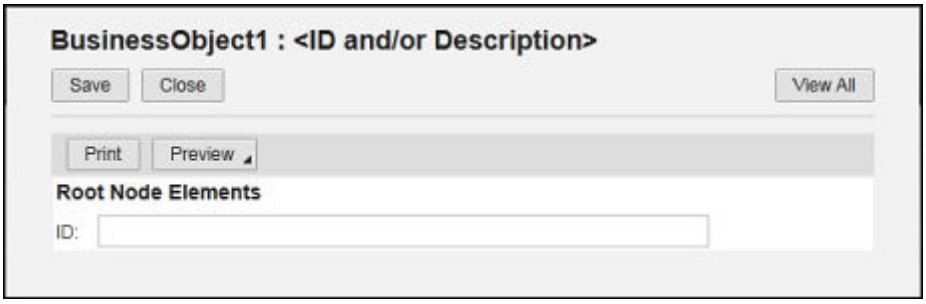


Bindings Configuration/Simple Navigation dialog box

4. On the *Bindings Configuration* tab, bind *Root* to *NodeReferenceBinding:*
   Click *Root* and *NodeReferenceBinding* and then click *Bind*.
5. On the *Simple Navigation* tab, right-click on *Navigations* and select *Add Navigation*.
6. In the *Navigation Details*, select the outport you created (for example, **`PreviewOutport`**) as *Outport* and select **`InPreview`** as *Inport*.
7. Bind each of the *Source Outport* parameters with the corresponding *Target Inport* parameter and save your changes.

4. **Create an event handler to fire the outport.**
   1. On the *Controller* tab, right-click *Event Handlers* and select *Add EventHandler*.
      The new event handler is displayed under *Event Handlers*.
   2. Right-click the new event handler and select *Rename*; then enter a new name for the event handler, for example, **`PreviewHandler`**.
   3. Under *EventHandler: Operations*, click the *Add Operation* button.
   4. Under *Type*, click in the field and then select *FireOutport*.
   5. Under *Configure the operation*, select the outport you have created (for example, **`PreviewOutPort`**) as *OutPort* and save your changes.
5. **To ensure that the user can use the preview button, associate the new event handler with the `OnClick` event.**
   1. On the *Designer* tab, select the navigation region.
   2. In the *Properties Explorer*, click in the *Toolbar* field and click *...* next to *Configure*.
      The *Configure Toolbar* dialog box opens
   3. Under *Available*, *StandardButtons*, select *Preview*.
   4. Add the button to *Current* by clicking the arrow pointing to the right.
   5. On the right, under *Events*, select *OnClick* and then the event handler you have created (for example, **`PreviewHandler`**) and click *OK*.
      The *Preview* button has been added to the navigation region.
   6. Save your changes.
6. **Test the Preview button in the SAP cloud solution.**
   1. Select the *Preview* tab next to the *Controller* tab.

2. Log on to the SAP cloud solution.
   The SAP cloud solution opens and the *Preview* button you have added is displayed as shown in the following figure.



The Preview button in SAP Business ByDesign

3. Click *Preview*.
   The *Preview* window opens.

Related Information

Print Forms Quick Guide [page 506]

9.4.3.5 Create a Pane in a Floorplan

Overview

You can create a pane to display different types of controls on a screen.

Prerequisites

You have created the floorplan to which you want to add the pane.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Select the pane, for example, the pane above which you want to add new fields.
   The pane is highlighted with a red border.
3. You have the following options:
   - To insert or delete a row or a column , right-click on the pane, select *Layout* and then choose the appropriate option.

- To expand the pane over two columns, click the arrow button on the right side of the pane.
- To move the pane to the next row, select the pane and drag the pane to the required row.

4. Drag and drop the required controls from the *Toolbox* into the new pane.
5. Bind each of the fields to the required business object element.
   For more information, see Bind Data Elements to Business Object Elements [page 731].
6. Save and activate your changes.

9.4.3.6 Create a Preview Pane

Overview

You can create a preview pane to preview information about a business object listed in a table.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. On the *Designer* tab, select the advanced list pane.
3. In the *Properties Explorer*, under *Misc – List Pane Variants*, click the arrow button.
   The *List Pane Variant Collection Editor* opens.
4. Under *Appearance*, set *Use Preview Pane* to `True`.
   The list pane selected, is now expanded with two rows. The system also allows you to add controls in those rows.
5. Drag the required controls from the *Toolbox Controls* to the preview panes.
   When you drag and drop a control, a *Section Group* is automatically created. You can add multiple controls to a section group and give a name to the section group. For example, the name of the section group can be Invoice Address.
6. In the *Properties Explorer*, under *Text Information – Field Group Header*, enter the required section group name.
7. Bind each of the fields to the required business object element.
   For more information, see Bind Data Elements to Business Object Elements [page 731].
8. Save and activate your changes.

Example

You have created an Object Work List (OWL) which shows a list of customers. You now want to create a preview pane, to view the following **invoice address details** of a selected customer, without leaving the OWL:

- Number
- Street
- City

- Pin Code
- Phone Number

You also want to view the number of orders that have not been shipped or delivered yet.

For each customer, you can configure a preview pane and add various fields to display the above information.

9.4.3.7 Change the Title of a Component

Overview

You can change the text of any component visible on the user interface. For example, you can change the following:

- Field label or tab text
- Group Header: Heading given to a group of fields.
- Roll Over Explanation Text: For field names that are not easy to understand, you can provide an explanation text. The system underlines fields with explanation texts in green and the explanation is visible on mouse over.
- Tooltip: In cases where a field name is abbreviated or is an icon, you can provide a tooltip for the field.
- Explanation Text: A sentence or two that is displayed on the user interface with details such as how to perform a task.
- Title of a floorplan, for example, a work center view.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Double-click the text that you want to change.
   The *Dependent Property Editor* dialog box opens.

   > **i Note**
   >
   > You can also change the text in the *Properties Explorer*, For example:
   >
   > - You can change the explanation text visible on the user interface under *Text Information – Explanation Area Text*.
   > - You can provide a rollover explanation text under *Text Information – Explanation*.

3. Enter the required text in the *Overidden text* field.

   > **i Note**
   >
   > All text options that can be changed for a particular component are displayed at the bottom of the *Dependent Property Editor* dialog box.

4. Click *OK*.

5. Save and activate your changes.

9.4.3.8 Hide a User Interface Component

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Select the UI component you want to hide, for example, a section group, or a field.
3. In the *Properties Explorer*, under *Behavior — Visible*, set the value to `False`.
4. Save and activate your changes.

Related Information

Change the Title of a Component [page 739]

9.4.3.9 Create a Text in the Text Pool

Overview

You can create text pools in order to collect all texts that you want to use for a particular user interface component, such as a floorplan.

The system-generated and the user-specific text are visible in the *TextPool* option in the *Controller* tab. The system-generated text is created by the tool. It is associated with a particular component, is marked in red and cannot be edited in the textpool.

The user- specific texts can be entered by the user and are mostly used in scripts. They are marked in black. You can reuse these texts when you are selecting a label for a control using the Reuse Text option. Based on the context, the texts with the correct text category are displayed on the user interface.

Prerequisites

You have created the floorplan for which you want to create the text pool entry.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. On the *Controller* tab, click *TextPool*.
3. Select the required language.
4. To add a new text to the text pool:
   1. Click *Add Text*.
      The UUID is automatically generated.
   2. Select a *Text Category* based on where you would like to reuse this text.
      X indicates short text that is less than 120 characters and **Y** indicates long text.
5. To enter the reusable text click *...* .
   The *Text Block Entry Editor* dialog opens.
6. In the *Text Block Entry Editor*, you can do the following:
   - **Edit** the static text: Change the text and click *OK*.
   - **Reuse** the static text: Click on *Reuse Text* and select one of the texts shown in the list and click *OK*.
   - **Add** an advanced concatenation of static and dynamic text parts. Dynamic parts inside the text are bound to data fields and are calculated during runtime.
     1. Click *Advanced*.
     2. To add a dynamic part to the text to the static text, click *...* in a new row.
     3. In the *Select Data Field* dialog, select the required data field and click *OK*.
7. Save and activate your changes.

9.4.3.10 Configure a Query

Overview

You can create and configure a query to view specific instances of a business object in a floorplan.

For example, in the *Contracts* OWL, the *Show* drop down list has entries such as **`Contracts in Preparation`**, and **`Released Contracts`**. These are the default sets under a query and are used to display a filtered set of contracts. You can create a query and default sets under that query. These default sets are then added to the drop down list.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Add a data field to create a reference to your data type, as follows:
   1. On the *Data Model* tab, right-click on root and then add a data structure.
   2. Right-click on the structure and then add a data field.

3. Configure the query, as follows:
    1. On the *Controller* tab, right-click on *Queries* and then choose *Add Query*.
    2. In the *Query Configuration*, select the required *Namespace* and *BO*.
    3. The queries modeled under the BO are displayed in a hierarchical manner.
       Expanding the query node in the tree will show all the query parameters.
    4. Select the required query and click *Bind*.
    5. Add the query parameters and select the data field that you created as the required value.
4. Configure the default set, as follows:
    1. Right-click on the query and then choose *Add Default Set*.
    2. To rename the default set, right-click on the default set and then choose *Rename*.
    3. In the *Default Set Configuration*, enter a *Title* for the default set.
       This value is displayed in the *Show* field.
    4. Select the query parameter you have added and define how you want the parameter to be used, by choosing the required *Select Options* and values.
       You can now use the default set in the OWL.
5. Select a default query, as follows:
    1. In the *Properties Explorer*, select the *Advanced List Pane*.
    2. Select the required default set, under *Misc — Default DefaultSet*.
    3. Under *Misc — Default Set Mapping*, click *...* .
    4. In the *Default Set Mapping* dialog box, move the default sets to the *Binding* area and click *OK*.
6. Save and activate your changes.

9.4.3.11 Reorder Columns in a List or a Table

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan is opened in the user interface designer.
2. Select the list.
3. In the *Properties Explorer*, under *Child Elements – List Columns*, click *...*.
4. In the *Column Collection Editor*, select the column you want to move.
5. Use the up and down arrows to move the column to the required position.
6. Save and activate your changes.

9.4.3.12 Configure an Object-Based Navigation in a Target Floorplan

Overview

You can use object-based navigation (OBN) to navigate between floorplans and pass parameters between floorplans.

The following instructions are to be performed in the floorplan that you would like to display on the click of a button. For example, the target floorplan can be the Product floorplan that is displayed when you click the Product ID link.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. On the *Data Model* tab, add a data field.
3. If required, rename the data field.
4. On the *Controller* tab, right click *Inports* and select *Add Inport*.
5. Check the *OBN Inport* check box.
6. Select the required namespace and business object (BO) model.
7. Under the BO, select the node for which the inport is being configured. If no node is selected, by default the root node will be chosen by the system.
8. Create an operation.
9. Select a *PortTypepackage* and *PortTypeReference* .
   The combination of business object, business object node, and the operation identifies the navigation target.
10. You can select one of the following port type references:
    - *OpenByKey*: Used at instance when you want to open the target floorplan for more information about an object in a read only mode. For example, the target floorplan is a customer fact sheet to view information about a customer.
    - *CreateWithRef*: Used at instance when you want to create an object with reference to the source floorplan. For example, the target floorplan is a Purchase Order OIF that is called from a Contract floorplan, to create a purchase order with reference to a contract.
11. Bind the data field you just created as the parameter for *Key*.
12. Bind the parent of the data field as the parameter for *KeyList*.
13. On the *Controller* tab, right-click on *Event handlers* and select *Add Eventhandler*.
14. If required, rename the event handler.
15. Select the Operations - Type as **`BOOperation`**.
16. Define the following *BO Operations Parameters*:
    1. Enter a name.
    2. Select the *Type* as **`keyNavigation`** .
    3. Select the data field that you created as *Bind*.
17. Select the inport you created.
18. In the *Properties Explorer*, under *Events – OnFire*, select the event handler you created.
19. Save and activate your changes.

Example

You are on the **Products** object work list (OWL) that displays a list of products, and you want to view the details of a selected product in the **Edit Product** OIF.

In the **Products** OWL, when a user does one of the following, you want to open the Edit Product OIF:

- Select a *Product ID*
- Selects a row and click *View* or *Edit*

To do this, you should configure an OBN in the source (OWL) and the target (OIF) floorplan.

9.4.3.13 Configure an Object-Based Navigation in a Source Floorplan

Overview

You can use object-based navigation (OBN) to navigate between floorplans and pass parameters between floorplans.

Perform the following steps in the floorplan where you click a UI control to open the target floorplan.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Right-click the user interface control that should open the target floorplan and choose *Add OBN using Wizard*.
3. The *Configure OBN* dialog box opens.
4. Under *Select BO details* , do the following:
   1. You can enter the name of the OBN being configured or the default name will be used.
   2. Select the namespace.
   3. Select the BO model.
   4. Select the node under the BO for which you want to configure the OBN. If no node is selected, the root node is chosen by default.
   5. Click *Next*.
5. Under *Select Operation*, do the following:
   1. From the list given, select the operation which needs to be executed. Each operation will have an associated *PortTypePackage* and *PortTypeReference* which get configured when you choose that particular operation. The combination of business object, business object node and the operation identifies the navigational target.
   2. Click *Next*.
6. Under *Configure Outport Data*, do the following:
   1. You can enter the name for the *Outport* being configured or the default name will be used.
   2. Add the *Context Attribute* details.
      Navigation can be made specific for a particular target floorplan by defining the context attributes, distinguishing it from a floorplan from the same business object, business object node and having the same operation configured for it.

3. Define the parameters you want to pass on to the target floorplan. The event handler is automatically created and bound to the outport.
   You can also rename the event handler which is being created.
4. Click *Finish*.

7. To edit a already configured OBN, right-click on the User Interface control for which the OBN was previously configured and choose *Edit OBN Using Wizard*.
8. Save and activate your changes.

Alternatives

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan is opened in the user interface designer.
2. Select the button that should open the target floorplan.
3. In the *Properties Explorer*, under *Menu Information – Navigation*, click ... .
4. In the *Model Navigation* dialog box, do the following:
   1. Select the target business object.
   2. Select the BO node defined in the target floorplan.
   3. Select the *Operation* defined in the target floorplan.
   4. Click *OK*.
5. On the *Controller* tab, select the outport that was automatically created.
6. Define the parameter you want to hand over to the target floorplan.
   The event handler is automatically created and bound to the outport.
7. To open the target floorplan on click of a link in a table, do the following:
   1. Select a column.
   2. In the *Properties Explorer*, select the *Display Type* as Link.
   3. Under *Events – OnClick*, define an event handler that is configured with FireOutport, to open the target floorplan.
8. Save and activate your changes.

9.4.3.14 Add a Link to Trigger an External URL

Overview

You can add a link that opens an external URL in your floorplans.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.

2. Drag the *Link* control from the *Toolbox* to the *Designer* tab.
3. On the *Designer* tab, select the *Link* user interface control.
4. In the *Properties Explorer*, under *Events - onClick*, select *New Event Handler*.
5. In the *Configure Event Handler* dialog box, select the *Operation Type* `OpenLink`.
6. Enter the required URL in the *Uri* field.
7. Click *OK*.
8. Save and activate your changes.

9.4.3.15 Write a Script

Overview

You can write front-end scripts while creating an event handler and while defining a calculation rule.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. In the *Designer* tab, select the user interface control to which you want to associate this script.
3. To write a script in an event handler, for example, to trigger an action on the click of a button, do the following:
   1. In the *Properties Explorer*, under *Events - onClick*, select *New Event Handler*.
   2. In the *Configure Event Handler* dialog, select the *Operation Type* as `Script`.
4. To write a script in a calculation rule, for example to control the visibility of a user interface control, do the following:
   1. In the *Properties Explorer*, under *Behavior – Visible*, select *Advanced*.
   2. In the *Dependent Property Editor*, select *Calculation Rule*.
5. Write the front-end script and click *OK*.
   In the script, you can:
   - Use the data elements defined in the *Data Model* tab.
   - Use the scope fields or the invalidation trigger fields to model the dependency between user interface controls.
   - Use the *IntelliSense* feature. For example you can select the data elements that are defined in the floorplan as you type the script.
   - Click *ZoomOut* to expand the editor for a bigger view.
   - Click *Check Syntax* to check if the script is syntactically correct.

   For information, see Front-End Script Reference [page 335].
6. Save and activate your changes.

9.4.3.16 Create an Event Handler

Overview

You can create an event handler to respond to user actions on the screen. For example, if you want to close the window when the user clicks on the *Close* button, you can create an event handler for the *Close* button. You can do either of the following:

- Create the user interface control, for example, a button, and then create an event handler.
- Create the event handlers first and then associate the event handler to the required user interface control.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Select the control for which you want to add an event handler.
3. In the *Properties Explorer*, under *Events – OnClick* select *New Event Handler*.
4. Enter a name for the event handler.
5. Select the type of operation.
   You can add more than one operation, for example, a `ListOperation` for an AddRow button, a `BOAction` for a Save button and a `WindowAction` for a Close button.
   For more information, see Operation Types [page 725].
6. Click *OK*.
7. Save and activate your changes.

Alternatives

You can also create an event handler first and then associate it with the required user interface control.

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan is opened in the user interface designer.
2. On the *Controller* tab, under the Tree view, right-click on *EventHandlers* and then choose *Add EventHandler*.
3. Enter a name for the event handler.
4. Select the type of operation.
5. On the *Designer* tab, select the UI control with which you want to associate the event handler.
6. In the *Properties Explorer*, under *Events – OnClick*, select the event handler you have created.
7. Save and activate your changes.

9.4.3.17 Configure an Action Form

Overview

You can create action forms to use the action parameters of a business object (BO) node or an enhanced controller object (ECO) node. Action forms are always bound to a BO/ECO node's ESI action that has parameters. The binding is done within the BO Action operation of an event handler. Action forms belong to a user interface component and cannot be reused in other user interface components.

You can either create an action form and then configure an event handler for it, or you can create an action form for an existing event handler.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. In the *Toolbox*, under *Panes*, drag *ActionForm* to the *Designer* tab.
3. Double-click the action form.
4. Set the following properties:
   - *Events – onCancelActionForm*: Create an event handler to handle the *Cancel* action in the action form. For more information, see Create An Event Handler [page 747].
   - *Misc– Title*: Enter a title for the action form.
   - *Misc – PreventDisplay*: It is usually set to `False` . You can display the action form, even if all the required parameters are not available at run time. In such a case, set it to `True`.
5. Drag the controls from the *Toolbox* to the action form.
6. Bind the controls to the fields which are bound to the action parameters.
7. Save and activate your changes.

Alternatives

You can also create an action form while configuring the event handler, which calls the corresponding ESI action.

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan is opened in the user interface designer.
2. In the *Controller* tab, select the event handler that triggers the ESI action.

   > **i** Note
   >
   > If there is no operation that calls an ESI action, create a new operation of type *BO Action* and map the operation to the corresponding ECO/BO action.

3. If the action parameters are not yet bound to data model attributes, bind these attributes in the *Selected Action Parameters* section.

4. Under *Action Form*, select the new action form.
   The action form with all the ESI action parameters will be created.
5. In the *Designer* tab, double-click on the action form.
6. If you want to display additional information, add the required fields.
7. Save and activate your changes.

9.4.3.18 Configure an Object Value Selector

Overview

You can configure an Object Value Selector (OVS) control to open an OVS dialog box.

Prerequisites

You have created the required OVS dialog box.

For more information, see Create an Object Value Selector Dialog Box [page 705].

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Do one of the following:
   - To add a new field, insert an *ObjectValueSelector* from the *Toolbox* controls.
   - To modify an existing field, select the required field, for example Employee. In the *Properties Explorer* under *Appearance*, set *DisplayType* to *ObjectValueSelector*.
3. In the *Properties Explorer*, under *Value Help - OVSComponent*, click *...* .
   The *Select OVS Component* dialog is displayed.
4. Click *Browse* to display the *Repository Browser*.
5. Select the *Target Component* from the tree.
6. If you want to reuse the SAP delivered OVS, click *Public OVS* and select an OVS component from the list.
7. Click *OK*.
8. Bind the ID attribute to the value.

   > **i Note**
   >
   > The CCTS type must be the same as defined in the outport of the OVS dialog box

9. Bind the description attribute to the text.

   > **i Note**
   >
   > The CCTS type must be the same as defined in the outport of the OVS dialog box

> The value for the text is not passed on automatically from the OVS box. You use the `AfterModify` script files to ensure that the texts are retrieved.

10. Save and activate your changes.

Example

You are creating a sales order and want to assign an employee responsible for that order. In the *Sales Order* floorplan, there is an *Employee* field, which is an OVS control. If you click this control, the OVS dialog appears where you can search for the required employee. The selected employee is displayed in the *Sales Order* floorplan.

9.4.3.19 Configure a Dedicated Field

Overview

You can add a dedicated field to a data model and use a field transformation to calculate the value of this dedicated field from existing business object (BO) node attributes.

For more information, see Transformations [page 724].

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. On the *DataModel* tab, right-click on Root and add a dedicated field.
3. On the *Controller* tab, right-click on *Field Transformations* and add a new *Field Transformation*.
4. For the *Field Transformation*, under *Target Field*, select the dedicated field that you created.
5. Select the *Namespace*.
6. Under *Transformation*, select the type of transformation rule you want to use.
   The parameters for that transformation rule are displayed. Each parameter section displays the parameter name and an indication if it is a mandatory field.
7. Each parameter can be configured by creating a data binding to the BO model or by assigning a constant value to it.
   - To configure a data binding, do the following:
     1. Select the *BO binding* checkbox.
     2. In the BO configuration area, click ....
     3. In the *Select Data Field* dialog box, select the BO node to be bound.
     4. Click *OK*.
   - To assign a constant value, select the *Constant* checkbox and, in the *Constant* field, enter a value.
8. Save and activate your changes.

9.4.3.20 Configure A Default Set Parameter Using Transformations

Overview

You can use a transformation to calculate the value of a default set parameter.

Different default sets are listed under a query.

For example, in the Contracts OWL, the Show drop down list has entries such as Contracts in Preparation, and Released Contracts. These are the default sets under a query and are used to display a filtered set of contracts. You can create a query and default sets under that query. These default sets are then added to the drop down list.

For more information, see Transformations [page 724].

You can have two types of transformations to calculate default set parameters:

- An example of the first type would be a transformation where no input parameters are needed. The transformation calculates the value needed using the login ID (`sy-uname`) and the `identity UUID` is the output.
- An example for the second type would be a transformation where the input parameters can only be constants 0 or 1 and the output is placed in the *Low Value* field of the *Default set* parameter row.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Add a data field, as follows:
   1. On the *Data Model* tab, right-click on Root and then add a data structure.
   2. Right-click on the structure and then add a data field.
3. Configure a query, as follows:
   1. On the *Controller* tab, right-click on *Queries* and then choose *Add Query*.
   2. In the *Query Configuration*, select the required *Namespace* and *BO*.
      The queries modeled under the BO are displayed in a hierarchical manner. Expand the query node in the tree to show all the query parameters.
   3. Select the required query and click *Bind*.
   4. Add the query *Parameters* and select the data field that you created as the required *Value*.
4. Configure the default set for the query, as follows:
   1. Right-click on the query and then choose *Add Default Set*.
   2. To rename the default set, right-click on the default set and then choose *Rename*.
   3. In the *Default Set Configuration*, enter a *Title* for the default set. This value is displayed in the *Show* field.
   4. Select the query parameter that you have added and click *Add Option*.
   5. In the new row that appears, under *Transformation* click *...*.
   6. In the *Configure Field Transformation* dialog box, select the *Namespace*.

7. Under *Transformation*, select the type of transformation rule you want to use.
   The parameters for that transformation rule are displayed. Each parameter section displays the parameter name and an indication if it is a mandatory field.
   Each parameter can be configured by assigning a constant value to it.
   For some transformation rules, there is no provision for entering input parameters.
8. In the *Constant* field, enter the value of the constant.

5. Save and activate your changes.

9.4.3.21 Add a Report List View to a Work Center

Overview

You need to assign standard report list views, for example the List and Gallery standard reports list views, to a work center in the user interface designer.

In the customer's production system (SAP cloud solution), an Analytics administrator assigns the report to the work center view to make it available for business users.

> **i** Note
>
> Only SAP cloud solution administrators working in a customer's production system can assign a report that you have created to a work center view. To assign a report to a work center view, the solution that contains the report must therefore be implemented in the customer's production system.
>
> We recommend that you provide information about the work center views to which customers need to assign reports in your Solution Documentation. For more information, see Documentation Types in Cloud Solutions from SAP [page 822].

Procedure

1. In the studio, double-click the required work center in the *Solution Explorer*.
   The work center opens in the user interface designer.
2. On the *Designer* tab, right-click on *BusinessObject* and select *WorkCenterItem*.
3. In the *Properties Explorer*, under *Misc-ViewSwitches* click *...*.
4. In the *ViewSwitch Collection Editor* dialog box, do the following:
   1. Click *Add* to add a new view.
   2. Select the newly added view.
   3. In the *Properties Explorer*, under *Misc-SubViewSwitches*, click *...*.
      The *SubViewSwitch Collection Editor* dialog box opens.
5. In the *SubViewSwitch Collection Editor* dialog box do the following:
   1. Click *Add* twice to add two new subviews.
   2. Click *OK*.
6. In the *ViewSwitch Collection Editor* dialog box, click *OK*.

7. On the *Designer* tab, select *WorkCenterView*.
8. Under *WorkCenterView Views*, select the first *WorkCenterSubView* that you added.
9. In the Configuration Explorer, navigate to *SAP_BYD_TF* > *Analytics* > *AnalysisPattern* > *ana_reports_view_wc.WCVIEW.uiwocview*.
10. Drag and drop this work center view onto the *Designer* tab.
11. In the dialog box that opens, click *Yes* to overwrite the work center view floorplan title with the view switch name.
12. Under *WorkCenterView Views*, select the second *WorkCenterSubView* that you added.
13. In the *Configuration Explorer*, navigate to *SAP_BYD_TF* > *Analytics* > *AnalysisPattern* > *ana_reports_carousel_view.WCVIEW.uiwocview*.
14. Drag and drop this work center view onto the *Designer* tab.
15. In the dialog box that opens, click *Yes* to overwrite the work center view floorplan title with the view switch name.
16. On the *Designer* tab, select *WorkCenterView*.
17. In the *Properties Explorer*, under *Misc-Text* click the arrow button.
18. In the *Dependent Property Editor* dialog box, do the following:
    1. Change the text in the *Overridden Text* field to `Reports`.
    2. Click *OK*.
19. Save and activate your changes.

Related Information

Create a Report [page 534]

9.4.3.22 Define an Anchor for a Floorplan

Overview

You can define anchors for your floorplans to allow end users to make changes to the floorplans provided by you.

Procedure

1. In the studio, double-click the required floorplan in the *Solution Explorer*.
   The floorplan opens in the user interface designer.
2. Select the region in floorplan for which you want to define the anchor.
   For example, you can define the anchor for the floorplan as a whole or for a section group, pane container, outport, and so on.
3. In the *Properties Explorer*, under *Extensibility–Anchors*, click ....

4. In the *Anchor Modelling* dialog box, click + to add an anchor.
5. Click *OK*.
6. Save and activate your changes.

> **i** Note
>
> You can view the anchors you have added only in the display mode.

9.4.3.23 Map a Message

Overview

You can map a message in order to change the terminology in messages appearing on the UI or to change their behavior on the UI. For more information see Message Mapping [page 754].

Procedure

1. In the *Solution Explorer*, click on the UI component and choose *Open in UI Designer*.
   The *UI Designer* opens with the selected UI component in a new window.
2. On the *Controller* tab, select *Messages –> System Messages*.
   The *Messages Editor* opens.
3. Select *Add Messages*.
   A new line in the *Message Mappings Editor* is displayed.
4. For the new message text that you want to be displayed on the UI instead of the original text do the following:
   1. In the *Message Key* column enter a key for the new message, instead of *<MESSAGETEX>*. The message key consists of alphanumeric characters.
   2. In the *Message Text* column enter the new message text for *<message text>*.
5. Save and activate your changes
6. Select *Messages –> Message Mapping*.
   The *Message Mapping Editor* opens.
7. Select *Add Messages* and in the first free line of the *Messages Editor* select ... to look for the message that you want to replace on the UI.
   A new window opens listing all the messages that are associated with the UI component.
8. Select the message that you want to replace and choose *OK*.
   The window closes and the message that you want to replace is displayed as a new line in the *Message Mappings Editor*.
9. In the *Message Mappings Editor*, select the rule that you want to apply in your message mapping.
   - To replace an original message with a new message, select *Replace*.
   - To keep the original message without any change, select *Original*.
   - To combine two or more original messages into one new single message, select *Aggregate* for each original message and select the same message key for each of them in the field *New Message Key*.
     .

- To prevent a message from being shown on the UI during runtime, select *Ignore*.

> **i Note**
>
> To show different mapped messages based on parameter values in one original message, proceed as follows:
>
> In the *Details* section of the *Message Mappings Editor*, in the field *Message Parameter*, select the parameter and in the field *Parameter Value*, enter the parameter value that you want the mapping to be based on.

10. On the *Details* screen in the field *New Message Text*, select the message text that you want to appear instead of the original message text.

> **i Note**
>
> If you want to aggregate two or more original messages, make sure that always select the same new message for those original messages.

11. Save and activate your changes.

9.5 Modify an SAP Floorplan

9.5.1 Change Transactions Quick Guide

This document gives you an overview of the changes you can make to SAP floorplans in the studio. These changes are termed as change transactions and need anchors to be executed.

Business and technical background

Change Transactions

Change transactions allow you to make changes to SAP delivered floorplans in the studio. You can create, save, and publish the change transactions using the Extensibility Explorer of the user interface designer, which is integrated into the studio.

Change transactions can only be made to SAP floorplans that contain stable anchors. For more information, see Anchors [page 759].

For more information, see here [page 758].

Anchors

Anchors are stable extension points to which any user interface modifications that you make to SAP floorplans are attached. This prevents life-cycle issues during updates or upgrades to the user interface. You can also add anchors to your own floorplans.

For more information, see here [page 759].

Tasks

Add a View to a Work Center

You can add a view to an SAP work center.

For more information, see here [page 761].

Hide a View from a Work Center

You can hide a view that is no longer required from an SAP work center.

For more information, see here [page 761].

Add an Assigned Object to a View

You can assign an object, for example, a navigation target or user interface component, to an SAP work center view.

For more information, see here [page 762].

Remove an Assigned Object from a View

You can remove unwanted objects from an SAP work center view.

For more information, see here [page 763].

Add an Embedded Component to an Overview View

You can add an embedded component such as an embedded report (ICP) to an SAP work center view of type "Overview".

For more information, see here [page 764].

Add an Embedded Component to a Floorplan

You can add an embedded component to an SAP floorplan.

For more information, see here [page 764].

Add an Embedded Component to a New Tab in a Floorplan

You can add an embedded component to a new tab in an SAP floorplan.

For more information, see here [page 766].

Add an Extension Field to a Screen

You can add an extension field to a screen. For more information see here [page 445].

Add an Extension Field to the Enterprise Search

You can add an extension field to the enterprise search. For more information see here [page 450].

Remove a Button

You can remove a button or navigational item from an SAP floorplan.

For more information, see here [page 766].

Modify the Order of Work Centers

You can modify the order in which the work centers are displayed in the **UI Settings XML** file.

For more information, see here [page 767].

Add a User Interface Dependency

User Interface (UI) dependencies govern the assignment of views to a user. Three types of dependencies between views can be maintained.

For more information, see here [page 768].

Save and Activate your Change Transactions

After you have made any of the changes listed above, you have to save and activate the change transactions.

1. Make your changes and then click *Apply/Add*.
   A list of unsaved change transactions is displayed in the *Unsaved Change Transactions* section of the *Extensibility Explorer*.
2. From the *File* menu, choose *Save*.
   A list of saved change transactions is displayed in the *Saved Change Transactions* section of the *Extensibility Explorer*.
3. From the *File* menu, choose *Activate*.
   In the *Activate Worklist* dialog box, select the check boxes beside the relevant components and click *OK*.
   The change transactions are activated and displayed in the *Published Change Transactions* section of the *Extensibility Explorer*.

9.5.2 Business and Technical Background

9.5.2.1 Change Transactions

Overview

Change transactions allow you to make changes to SAP delivered floorplans in the studio. You can create, save, and publish the change transactions using the Extensibility Explorer of the user interface designer, which is integrated into the studio.

Change transactions can only be made to SAP floorplans that contain stable anchors. For more information, see Anchors [page 759].

When you create change transactions they are stored in the associated solutions created by you.

You can use a change transaction to add a view to a work center delivered by SAP. For example, you can add a new view to the Liquidity Management work center (CashLiquidityManagement_WCF).

9.5.2.2 Anchors

Overview

Anchors are stable extension points to which any user interface modifications that you make to SAP floorplans are attached. This prevents life-cycle issues during updates or upgrades to the user interface. You can also add anchors to your own floorplans.

Any user interface adaptations made by you, as an SAP partner, to SAP floorplans, are retained unchanged in future releases even after the upgrades provided by SAP due to the presence of anchors.

Features

A stable anchor has a system-wide unique ID and a reference entity. All stable anchors created during a particular release will also exist in later releases.

Anchors are essential when making changes to a floorplan that is from a layer lower than the layer you are working on. You can only view the anchors added in a lower layer. For example, when you are working on a SAP floorplan you can view the anchors provided by SAP. You can view the anchors added by you only when a floorplan is opened in the display mode.

Stable anchors in the floorplans are defined for a logical set of fields placed on the user interface, for example a section group or a list.

Use

All change transactions can only be applied if there is a stable anchor defined on the SAP floorplan.

Requirement	Required anchor	Link to document
Add a View to a Work Center	WoCViewSwitchAnchor	Add a View to a Work Center [page 761]
Hide a View From a Work Center	WoCViewSwitchAnchor	Hide a View from a Work Center [page 761]
Add an Assigned Object to View	AssignedObjectAnchor	Add an Assigned Object to View [page 762]
Remove an Assigned Object from View	AssignedObjectAnchor	Remove an Assigned Object from View [page 763]
Add an Embedded Component to an Overview Page	OverviewPageAnchor	Add an Embedded Component to an Overview View [page 764]

Add an Embedded Component to a Floorplan	OutportAnchor and PaneContainerAnchor	Add an Embedded Component to a Floorplan [page 764]
Add an Embedded Component to a New Tab in a Floorplan	PaneContainer Anchor or ViewSwitchNavigationAnchor or TabStripAnchor	Add an Embedded Component to a New Tab in a Floorplan [page 766]
Remove a Button	ButtonGroupAnchor	Remove a Button [page 766]
Modify the Order Of Work Centers	WorkCenterListAnchor	Modify the Order of Work Centers [page 767]
Adding a User Interface Dependency	UIDependecyAnchor	Add a User Interface Dependency [page 768]
Add a Mashup to a Floorplan	OutportAnchor FloorPlanAnchor. OverviewPageAnchor	Add a Mashup to Your Floorplan [page 568]

Related Information

Define an Anchor for a Floorplan [page 753]

9.5.2.3 Flexible UI

Overview

In the SAP cloud solutions, a UI is modeled generically and not for a specific user or specific user groups, therefore it looks the same for all users of a solution. Sometimes it might be necessary to simplify or reduce the appearance of a view at runtime so that it shows, for example, fewer fields or sections depending on which user or user group is working with a UI. UI switches in the UI designer allow to hide or restrict the usage of fields on one or more UIs. Note that you can use UI switches only to restrict the usage or appearance of fields on a UI that has been developed by SAP. You cannot use UI switches, for example, to change read-only fields into fields that can be edited.

For more information see Create a Role-Based Flexible UI Using a UI Switch [page 772].

It is also possible to change the behavior of fields depending on the status of other fields.

For more information see Example: Create a Status-Based Flexible UI [page 770].

9.5.3 Tasks

9.5.3.1 Add a View to a Work Center

Overview

You can add a view to an SAP work center.

Prerequisites

The SAP work center to which you want to add a view has a *WoCViewSwitchAnchor*. For more information, see Anchors [page 759].

Procedure

1. In the studio, open the user interface designer.
2. In the *Configuration Explorer*, navigate to the work center where you want to add the view.
3. Double-click the work center.
4. In the *Extensibility Explorer*, select *WoCViewSwitchAnchor*.
5. Click *Add* and then choose *AddWoCView*.
6. In the *Add Work Center View* dialog box, click *Target Component* to select the required view. You can only select components of type **`uiwocview`** (work center view) and subtype **`Application`**.
7. Select the required position:
   - *Top* places the new view as the first view in the work center. If an Overview view exists in the work center, then the new view is positioned after the Overview view.
   - *Bottom* places the new view as the last view in the work center. If a Reports view exists, then the new view is positioned before the Reports view.
8. If you want to control the scope dependency of the view in the work center, click *Add Scoping Rules*.
9. In the *Scope Rules Editor* dialog box, choose the scoping rule from the list on the left and use the arrow buttons to move it to the list on the right.
10. Click *OK*, to apply your changes to the view.
11. Click *Apply*.
12. Save and activate your changes.

9.5.3.2 Hide a View from a Work Center

Overview

You can hide a view that is no longer required from an SAP work center.

Prerequisites

- The work center has a *WoCViewSwitchAnchor*.
  For more information, see Anchors [page 759].

Procedure

1. In the studio, open the user interface designer.
2. In the *Configuration Explorer*, navigate to the work center in which you want to hide a view.
3. Double-click the work center.
4. In the *Extensibility Explorer*, select *WoCViewSwitchAnchor*.
5. Click *Add* and then choose *HideWoCView*.
6. In the *Hide Work Center View* dialog box, select the title of the view you want to hide.
7. Click *Apply*.
8. Save and activate your changes.

9.5.3.3 Add an Assigned Object to a View

Overview

You can assign an object, for example, a navigation target or user interface component, to an SAP work center view.

Prerequisites

The SAP work center view, to which you want to assign an object, has an *AssignedObjectAnchor*.

For more information see, Anchors [page 759].

Procedure

1. In the studio, open the user interface designer.
2. In the *Configuration Explorer*, navigate to the work center view where you want to assign the object.
3. Double-click the work center view.
4. In the *Extensibility Explorer*, select *AssignedObjectAnchor*.
5. Click *Add* and then choose *AddAssignedObjectToView*.
6. In the *Add Assigned Object* dialog box, select the type of object you want to assign.

7. Click *Target Component* to select the object.
8. Depending on the type of assigned object, you have the following options:
    - If the assigned object is of type **Navigation Target**, you can add scoping rules:
        1. Click *Add Scoping Rules*.
        2. In the *Scope Rules Editor* dialog box, choose the scoping rule from the list on the left.
        3. Use the arrow buttons to move the selected scoping rule to the list on the right. Click *OK* to apply your changes.
    - If the assigned object is an object work list, then you can mark the view as *Expose to Overview*. The object work list will then be visible in the Overview view.
9. Click *Apply*.
10. Save and activate your changes.

9.5.3.4 Remove an Assigned Object from a View

Overview

You can remove unwanted objects from an SAP work center view.

Prerequisites

The work center view has an *AssignedObjectAnchor*.

For more information, see Anchors [page 759].

Procedure

1. In the studio, open the user interface designer.
2. In the *Configuration Explorer*, navigate to the work center view in which you want to remove an assigned object.
3. Double-click the work center view.
4. In the *Extensibility Explorer*, select *AssignedObjectAnchor*.
5. Click *Add* and then choose *RemoveAssignedObjectToView*.
6. In the *Remove Assigned Object* dialog box, select the object you want to remove.
7. Click *Apply*.
8. Save and activate your changes.

9.5.3.5 Add an Embedded Component to an Overview View

Overview

You can add an embedded component such as an embedded report (ICP) to an SAP work center view of type "Overview".

Procedure

1. In the studio, open the user interface designer.
2. In the *Configuration Explorer*, navigate to the Overview view where you want to add an embedded component.
3. Double-click the floorplan.
4. In the *Extensibility Explorer*, select the *OverviewPageAnchor*.
5. Click *Add* and then choose *AddContenttoOverview*.
6. In the *Add Component* dialog box, do the following:
    1. Select the floorplan type, for example, an object work list.
    2. Click *...* under *Component* and select the floorplan.

        > **i** Note
        >
        > If you select the floorplan type as ICP, select the required report.

    3. Select the required *Title*, *Position* and *Alignment* of the embedded component.
7. If the embedded component is of the generic type ICP, you have to select the parameters associated with the ICP under the option *View*.
8. Click *Add*.
9. Save and activate your changes.

9.5.3.6 Add an Embedded Component to a Floorplan

Overview

You can add an embedded component to an SAP floorplan. You can add the embedded component to either an *OutportAnchor* or a *PaneContainerAnchor*.

Prerequisites

- The floorplan has an *OutportAnchor*.
  For more information, see Anchors [page 759].

- The floorplan has a *PaneContainerAnchor*.
  The embedded component (EC) will be placed behind the pane container.
- The embedded component has an inport.
  The *RequestAutoRefire* and *RequestFireOnInitialization* properties of the embedded component are set to `True`.

Procedure

1. In the studio, open the user interface designer.
2. In the *Configuration Explorer*, navigate to the floorplan in which you want to add an embedded component.
3. Double-click the floorplan.
4. In the *Extensibility Explorer*, do one of the following:
   - To add the embedded component to an *OutportAnchor*:
     1. Select the *OutportAnchor*, click *Add* and then choose *Add Embedded Component*.

        > **i** Note
        >
        > The outport selected has an *OutportAnchor* and the *PaneContainerAnchor* set as the referenced anchor.

     2. In the *Add Embedded Component* dialog box, click *Select Embedded Component* and select the required embedded component.
     3. In the *Add Embedded Component* dialog box, if the referenced pane container for which the *PaneContainerAnchor* has been defined, contains a tab strip, you can enter a *Tab Title*.
        1. At runtime, a new tab is created with this title, and the embedded component appears in the tab.
        2. If the referenced pane container does not contain a tab strip, the embedded component appears in a new row below the referenced pane container.
     4. Click *Define Mapping* and select the inport of the embedded component.
     5. If the inport has any parameters, map the parameters.
     6. If the embedded component is of the generic type ICP, you are prompted to provide a title for the ICP.
   - To add the embedded component to a *PaneContainerAnchor*:
     1. Select the *PaneContainerAnchor*, click *Add* and then choose *Add Embedded Component*.
     2. In the *Add Embedded Component* dialog box, if the referenced pane container, for which the *PaneContainerAnchor* has been defined, contains a tab strip, you can enter a *Tab Title*.
        1. At runtime, a new tab is created with this title, and the embedded component appears in the tab.
        2. If the referenced pane container does not contain a tab strip, the embedded component appears in a new row below the referenced pane container.
     3. Click *Select Embedded Component* and select the required embedded component.
     4. If you want to bind the embedded component to the floorplan, click *Bind*.
     5. In the dialog box that opens, select the outport and inport for the embedded component. The outport selected has an *OutportAnchor*.
5. Click *Apply*.
6. Save and activate your changes.

9.5.3.7 Add an Embedded Component to a New Tab in a Floorplan

Overview

You can add an embedded component to a new tab in an SAP floorplan.

Prerequisites

- The floorplan has either the *ViewSwitchNavigationAnchor* or the *TabStripAnchor*.
  For more information, see Anchors [page 759].
- The embedded component has an inport.

Procedure

1. In the studio, open the user interface designer.
2. In the *Configuration Explorer*, navigate to the floorplan in which you want to add an embedded component.
3. Double-click the floorplan.
4. In the *Extensibility Explorer*, select the *ViewSwitchNavigationAnchor* or the *TabStripAnchor*.
5. Click *Add* and then choose *Add Embedded Component*.
6. In the *Add Embedded Component* dialog box, enter a *Tab Title*.
7. Click *Select Embedded Component* and select the required embedded component.
8. If you want to bind the embedded component to the floorplan, click *Bind*.
9. In the dialog box that opens, select the outport and inport for the embedded component. The outport selected has an *OutportAnchor* associated with it.
10. Click *Apply*.
11. Save and activate your changes.

9.5.3.8 Remove a Button

Overview

You can remove a button or navigational item from an SAP floorplan.

Prerequisites

The SAP floorplan from which you want to remove a button or navigational item has a *ButtonGroupAnchor*.

For more information, see Anchors [page 759].

Procedure

1. In the studio, open the user interface designer.
2. In the *Configuration Explorer* navigate to the required floorplan.
3. Double-click the floorplan.
4. Select a button group in the floorplan.
5. In the *Extensibility Explorer*, select the *ButtonGroupAnchor*.
6. Click *Add* and then choose *RemoveButton*.
7. In the *Remove Button* dialog box, select a button or navigational item that you want to remove from the floorplan.
   If the button or navigational item cannot be removed, the *Apply* button is disabled.

   > **i** Note
   >
   > The removal of mandatory buttons like **Cancel**, **Close**, and **Print** is restricted.
8. Click *Apply*.
9. Save and activate your changes.

9.5.3.9 Modify the Order of Work Centers

Overview

You can modify the order in which the work centers are displayed in the **UI Settings XML** file, the path to which is given below.

Prerequisites

The UI Settings file has a *WorkCenterListAnchor*.

For more information, see Anchors [page 759].

Procedure

1. In the studio, open the user interface designer.
2. In the *Configuration Explorer*, go to *SAP_BYD_UI_CCOW* > *Main* > *root* > *uisetting* .
3. Double-click the file.

4. In the *Extensibility Explorer*, select the *WorkCenterListAnchor*.
5. Click *Add* and then choose *WorkCenterReordering*.
6. You can do the following in the *Work Center Reordering* dialog box:
    - Add work centers to the list of *Ordered WorkCenters*.
        - Select the required work center from the list of *Unordered WorkCenters* and use the right arrow button to add the work center to the list of *Ordered WorkCenters*.
          The required work center is added to the list of *Ordered WorkCenters*, below a work center selected in the list.
    - Remove work centers from the list of *Ordered WorkCenters*.
        - Select the required work center from the list of *Ordered WorkCenters* and use the left arrow button to move the work center to the list of *Unordered WorkCenters*.
    - Change the position of the work center.
        - Select a work center and use the up and down arrow buttons to change its position in the list of *Ordered WorkCenters*.
        - To place the selected work center at the top of the list of *Ordered WorkCenters* click *Top*.
        - To place the selected work center at the bottom of the list of *Ordered WorkCenters* click *Bottom*.
7. Click *Apply*.
8. Save and activate your changes.

9.5.3.10 Add a User Interface Dependency

Overview

User interface (UI) dependencies govern the assignment of views to a user. Three types of dependencies between views can be maintained.

Types of Dependencies

Dependency	Description
Prerequisite	Ensures that if one work center view is assigned to a user, then the other work center view in the dependency is automatically assigned to the user.
Technical Conflict	Ensures that the user is not assigned two work center views that conflict with each other from a technical point of view. When there is a technical conflict, an error message is displayed and assignment of views is not allowed . For example, if both the Postings and Master Data views of the Fixed Assets workcenter are assigned to a user, then an error message appears and the assignment is not carried out.

SoD Conflict (Segregation of Duty)	Ensures that the user is not assigned two views (chosen while defining the dependency) that conflict with each other from a best practice point of view. For example, if the user has two views assigned that enable the user to both place an order and to approve the order. These functions should be performed by different users. When a user assignment generates a SoD conflict, a warning message is displayed .

These user interface dependencies will govern the assignment of views in the User and Access management view. For more information, see the relevant documentation in the SAP cloud solution.

The existing dependencies are listed in the **UIDependency XML** file, the path to which is given below.

Prerequisites

The **UIDependency XML** file has a *UIDependencyAnchor*.

For more information, see Anchors [page 759].

Procedure

1. In the studio, open the user interface designer.
2. In the *Configuration Explorer*, go to *SAP_BYD_UI_CCOW* > *Main* > *root* > *uidependency* .
3. Double-click the **UIDependency XML** file.
4. In the *Extensibility Explorer*, select the *UIDependencyAnchor*.
5. Click *Add* and then choose *UI Dependency*.
6. You can add three kinds of UI dependencies: *Prerequisite*, *Technical Conflict* , and *SoD Conflict*.
   - If you want to add a **prerequisite dependency**:
     1. In the *UI Dependency* dialog box, click the *Prerequisites* tab.
     2. Click + to add a new prerequisite.
     3. Select the desired views for *Target Partner WoC View* and *Prerequisite WoC View*.
   - If you want to add a **technical conflict dependency**:
     1. In the *UI Dependency* dialog box, click the *Technical Conflict* tab.
     2. Click + to add a new technical conflict.
     3. Select the desired views for *Target Partner WoC View* and *Technical Conflicting WoC View* .
     4. In the *Reason* field, enter the reason why a user should not be assigned both views at the same time.
   - If you want to add a **SoD conflict dependency**:
     1. In the *UI Dependency* dialog box, click the *SoD Conflict* tab.
     2. Click + to add a new SoD conflict.
     3. Select the desired views for *Target Partner WoC View* and *SoD Conflicting WoC View* .
     4. In the *Reason* field, enter the reason why a user should not be assigned both views at the same time.

5. In the *Resolution* field, enter the solution to the issue.

7. Click *Apply*.
8. Save and activate your changes.

9.5.3.11 Example: Create a Status-Based Flexible UI

Overview

The following section provides a detailed example of how to create a business object extension of the business object Opportunity and the corresponding UI that shows an additional field depending on the status of the field value.

In the example, the business object extension contains the additional field `Reason Text`, which is shown on the UI when an opportunity exceeds a certain value. The user must enter a text in the field `Reason Text` in order to be able to save the opportunity. If the user does not enter anything in the field and tries to save the opportunity, the UI displays an error message and the user cannot save the opportunity. If the value of the opportunity is below the stated threshold, the field *Reason Text* is not displayed on the UI.

Prerequisites

You have created a solution.

Procedure

1. In the *Solution Explorer*, expand your solution, right-click your project file, and select *Add* > *New Item*. The *Add New Item* dialog appears.
2. Select *Business Object Extension*.
3. Enter a *Name* for your business object extension, in our example it is `OpportunityExtension` and click *Add*.

   > **i Note**
   >
   > You cannot use a name that is already used for a business object or business object extension in your solution.

   The *Business Object Selection* dialog appears.
4. Select the namespace, in our example `http://sap.com/xi/AP/CRM/Global`, the business object that you want to extend, in our example, `Opportunity`, and select *OK*.
   The business object extension item with the name `OpportunityExtension.xbo` is added to your project. This file contains a template for the business object extension, including the extendable nodes.
5. In the *Solution Explorer*, double-click the business object extension to open the file in the editing pane.

6. Enter the required coding in the business object extension. In our example, the coding consists of an element `Reason`, an element `IsBig`, which serves as an indicator, and a message that is displayed on the UI, when a certain value of an opportunity is exceeded.

```
import AP.Common.GDT;
import AP.CRM.Global;
[Extension] businessobject AP.CRM.Global:Opportunity
  raises MessageEnterReason
{

  // You must activate this business object before you can access the
extension fields
   // or messages in script files, forms, and screens.

  message MessageEnterReason text "Enter reason for 'big' opportunity";
  [Label("Reason text")] element Reason : LANGUAGEINDEPENDENT_EXTENDED_Text;
  element IsBig : Indicator;
node Item {
}
node Party {
}
}
```

7. Save and activate the business object extension.
8. In the *Solution Explorer*, right-click the business object extension and select *Create Script Files*.
   The *Create Script Files* dialog window opens.
9. Scroll to the `SalesForecast` node, in the *Create* column, set the checkbox for `Event:AfterModify`, and click *OK*.
   The `Event-AfterModify.absl` object is created and listed in the *Solution Explorer*.
10. Enter the necessary ABSL coding. In our example, the threshold for the opportunity value is 100000.

```
import ABSL;
```

```
if (this.ExpectedRevenueAmount.content > 100000) {
       this.ToRoot.IsBig = true;
 }
 else {
       this.ToRoot.IsBig = false;
 }
```

11. In the *Solution Explorer*, double-click the *Event-AfterModify.absl* to open the file in the editing pane.
12. Enter the necessary ABSL coding. In our example, the threshold for the opportunity value is 100000.

```
import ABSL;
```

```
if (this.ExpectedRevenueAmount.content > 100000) {
         this.ToRoot.IsBig = true;
 }
 else {
         this.ToRoot.IsBig = false;
 }
```

13. Save and activate your changes.
14. In the *Solution Explorer*, right-click the business object extension and select *Enhance Screen*.
    The *Enhance Screen* dialog window opens.
15. Select the required floorplan, in our example, the *COD_Opportunity_TI* and select *OK*.
    The `.xuicomponent` file is created and added to your solution.
16. In the *Solution Explorer*, right click the `.xuicomponent` and select *Open in UI Designer*.

17. Select the header, in this example the second header, and in the *Extensibility Explorer*, select *Add Extension Field to Section Group*.
    The *Add Extension Field* dialog window opens.
18. In the *Add* column, select the `Reason text` and select e *Apply*. Save and Activate your changes.
    The field *Reason Text* in now visible on the UI.
    Save and activate your changes.
19. Select the second header again, and in the *Extensibility Explorer*, select *Adjust Properties*.
    The *Adjust Properties* dialog window opens.
20. Select the *Reason text* control and in the *Visible* column, select *Bound*.
    The *BO Field Selector* dialog window opens.
21. In the *BO Field Selector* dialog, select `IsBig` and click *OK*. The dialog window closes.
22. In the *Adjust Properties* dialog window, select the *Reason text* control again and in the *Mandatory* column, select *Bound*.
23. The *BO Field Selector* dialog window opens.
24. In the *BO Field Selector* dialog window, select `IsBig` and click *OK*.
    The *BO Field Selector* dialog window closes.
    In the *Adjust Properties* dialog window, select *Apply*. Save and activate your changes.
25. In the *Solution Explorer*, right-click the business object extension and select *Create Script Files*.
    The *Create Script Files* dialog window opens.
26. Scroll to the `Root` node, in the *Create* column, set the checkbox for `Validation:OnSave`, and click *OK*.
    The `Validation—OnSave.absl` object is created and added to your solution.
27. In the *Solution Explorer*, double-click the `Validation-OnSave.absl` file to open it in the editing pane and enter the following coding:

```
if (this.SalesForecast.ExpectedRevenueAmount.content > 100000 &&
this.Reason.IsInitial()) {
  raise MessageEnterReason.Create("E");
      return false;
```

```
}
```

```
 else {
  return true;
```

    Save and activate your changes.

9.5.3.12 Create a Role-Based Flexible UI Using a UI Switch

Overview

In the UI designer it is possible to hide or display extension or standard fields on the UI. It is also possible to make them read-only or mandatory depending on your requirements. To do this, you can create an UI switch in the studio.

> **i** Note
>
> With an UI switch you can make the fields on the UIs provided by SAP more restrictive, but you cannot make them less restrictive.

Prerequisites

You have created a solution.

If you want to hide or display extension fields on the UI depending on the role that is assigned to a user, you have created a business object extension and the fields that you require. You have added those fields to the section of the UI where you want them to be shown.

Procedure

1. In the *Solution Explorer*, expand your solution, right-click your project file, and select *Add* > *New Item*.
   The *Add New Item* dialog appears.
2. Select *UI Switch*.
3. Enter a *Name* for the UI switch and click *Add*.

   > **i Note**
   >
   > You cannot use a name that is already used for a UI switch in your solution.

   The *UI Switch Description* dialog window appears.
4. Enter a description for the UI switch and select *OK*.
   The UI switch is added to your project.
5. In the *Solution Explorer*, double-click the UI component where you want to use the UI switch to, for example, hide or display fields based on a role.
   The UI component opens in the UI designer.
6. In the *Extensibility Explorer*, select the entity of the UI where you want to, for example, hide or display fields and select *Adjust Properties*.
   The *Adjust Properties* dialog window opens.
7. In the *Adjust Properties* dialog window, you have the following options:
   - If your settings are to apply to all roles that the UI switch is assigned to, for the *UI Switch* select *Equal* and the UI switch that you have created.
   - If your settings are to apply to all roles that the UI switch is not assigned to, for the *UI Switch* select *Not Equal* and the UI switch that you have created.
   - The following settings do not have any effect on how UI is displayed:
     - *Equal* and *None*
     - *Not Equal* and *None*
8. For any control on the UI, you have now the following options:
   - Column *Visible*:
     - If you want the field and its behavior on the UI to be unchanged, select *None*
       If you want to hide the field on the UI, select *False*.
       If you want the field to be displayed depending on an indicator, select *Bound* and in the *BO Field Selector* dialog window, the corresponding indicator.
       If you want the field to be displayed in the case of a role that has 2 UI switches assigned, select *True*.
   - Column *Mandatory*:
     - If you want the field and its behavior on the UI to be unchanged, select *None*

If you want the field to be a mandatory field (that means, a field where the user has to enter something), select *True* .

> **Note**
>
> If you set a field to be mandatory, this does not automatically prevent saving or any other actions. It only marks the field as mandatory on the UI. In addition, you must create a message in the BO extension file (`.xbo`) and raise the message in the related script coding (`Event-AfterModify.absl` file or `Validation-OnSave.absl` file).

If you do not want the field to be a mandatory field, select *False*.
If you want the field to be mandatory depending on another field, select *Bound* and in the *BO Field Selector* dialog window, the corresponding indicator.

- Column *Read Only*:
  If you want the field and its behavior on the UI to be unchanged, select *None*.
  If you want the field to be read only, select *True*.
  If you want the field not to be read-only, select *False*.
  If you want the field to be read-only depending on another field, select *Bound* and in the *BO Field Selector* dialog window, the corresponding indicator.

9. Select *Apply*.

> **Note**
>
> To apply the UI switch to the UI, you have to assign it to a business role and assign this business role to the users.
>
> To read out the UI switches of a user, you can use the function `Context.GetCurrentUserUISwitches()` in the script coding.

9.6 User Interface Modifications for Mobile Applications

9.6.1 Mobile Solutions

Overview

You can create solutions in the studio and make them available on mobile devices, such as the iPhone and Android devices. The mobile solution is built using the same tools (the studio and UI designer) that you use for the browser application of the SAP cloud solution. You just have to make sure that you build your mobile application according to the user interface (UI) guidelines for mobile applications (for example, tables should not have more than three columns).

Supported Platforms

The mobile application is currently available for:

- Android phones
- Apple iPhone

Device Tag

You can use the *Device* tag in the UI designer to specify on which devices a UI component can run. Possible values are:

- *Desktop*
- *Mobile*
- *Tablet*
- *Mobile HTML*
- *All Devices*

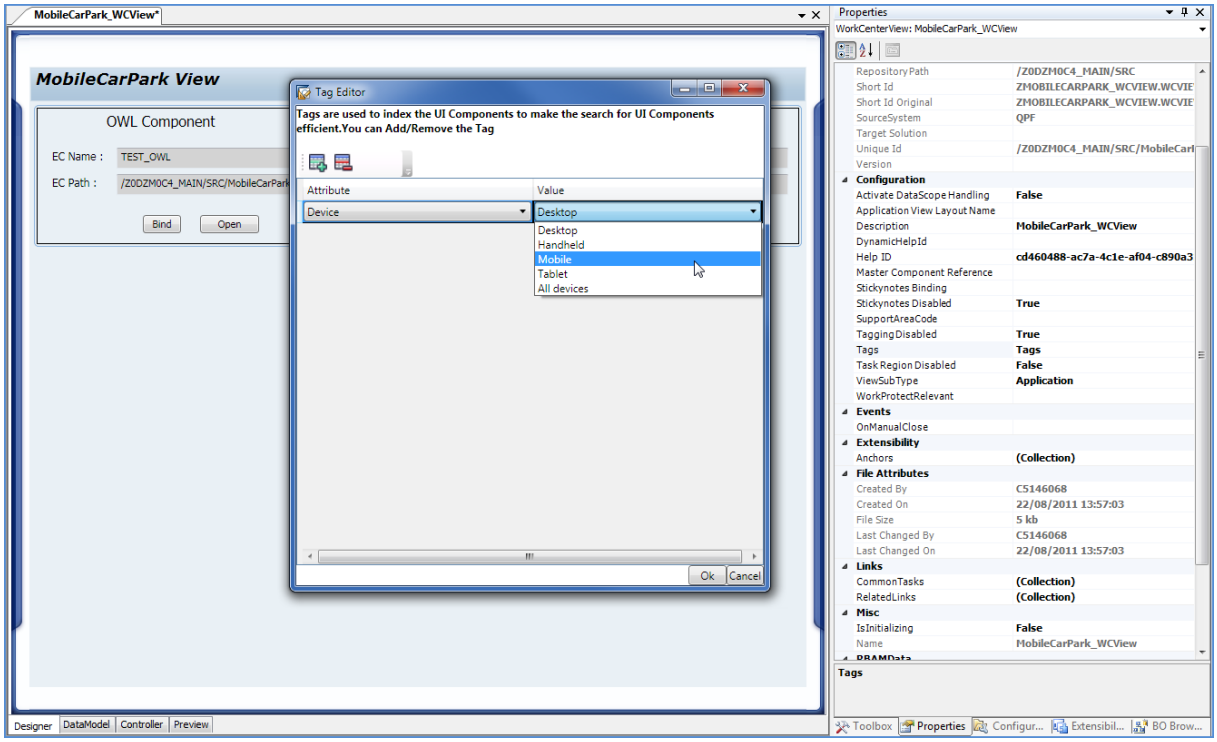
If no device tag is specified, the component is treated as having a *Desktop* device tag.

The following components can be tagged:

- work center views
- flooplans
- reusable components (such as an Object Value Selector)

You must consider the dependencies between floorplans when setting the *Device* tag. For example, you cannot set a work center view (`WCV`) to *Mobile* if it has an embedded object work list (`OWL`) that is set to *Desktop*. In this case, you would first have to set the `OWL` to *Mobile*, then set the `WCV` to *Mobile*.

Floorplan with Device tag	Can use floorplan with Device tag
Desktop	All Devices or Desktop or no device tag
Mobile	All Devices or Mobile
Tablet	All Devices or Tablet
All Devices	All Devices
no device tag	All Devices or Mobile HTML or Desktop or no device tag

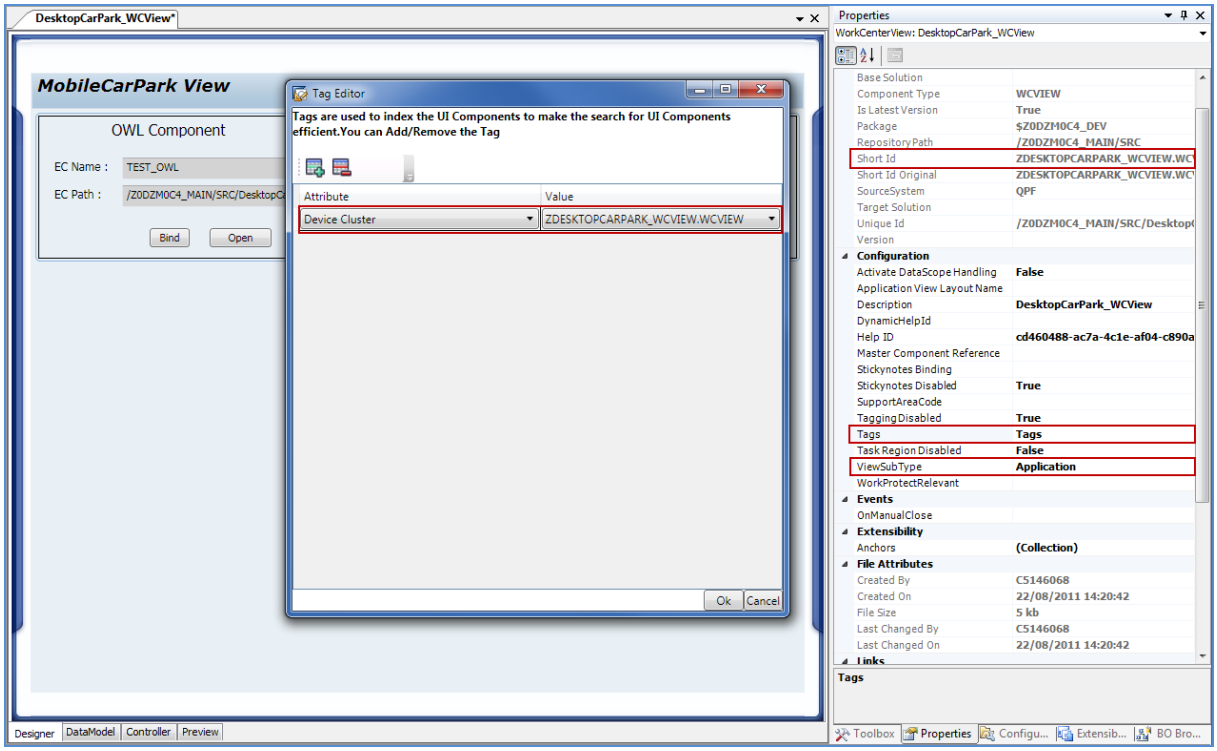


Device Cluster Tag

If you have more than one view of a work center (for example, a desktop view and a mobile view), you can use the *Device Cluster* tag in the UI designer to model the relationship between the views. The user assignment takes place on the leading work center view, which is always the desktop view. The view to render will be decided at runtime based on the user client.

The leading work center view (`WCV`) has to have a *ViewSubType* of *Application*. You then create a *Device Cluster* and set the value to the *Short Id* of the `WCV`. To associate a mobile view, you give it a *ViewSubType* of *DeviceView* and assign it to the Desktop Workcentre View *Device Cluster*.

The other possible value for the *ViewSubType* is *InvisibleDeviceView*. You would select this if you did not want the view to be displayed on the UI, but wanted the view to serve as a carrier of authorizations only.



Mobile Scenarios

There are a number of scenarios to consider when developing a mobile solution.

- **Mobile Only — SAP Business Objects**
  You can create a mobile solution, without a desktop variant, that displays a view of information that exists in the SAP business objects (BOs). In this case, you could either create the screens individually from *Solution Explorer* > *Add* > *New Item* > *Screens* and then configure the navigation in the UI designer, or you could create a blank BO, generate screens, and then edit the screens in the UI designer. If there is no desktop variant, the *ViewSubType* of the `WCV` must be set to *Application* or *InvisibleApplication* (based on the use-case).
- **Mobile Only — Partner Business Objects**
  In this case, you would generate the screens as normal in the SDK and configure the screens in the UI designer. If there is no desktop variant, the *ViewSubType* of the `WCV` must be set to *Application* or *InvisibleApplication* (based on the use-case).
- **Mobile and Desktop Solution — Same Screens**
  In this case, you can set the device tag to *All Devices* for all of your screens.
- **Mobile and Desktop Solution — Different Screens**
  In this case, you can create two sets of floorplans, one for desktop and one for mobile. You can then delete the mobile `WCF`, create a *Device Cluster* with the *Short Id* of the desktop `WCV`, then add the mobile `WCV` to the *Device Cluster*.

Testing your Mobile Solution

You can view the mobile variant of the SAP solution by appending `&sapbyd-agent=mobile` to the standard URL. This will show you which work centers and floorplans are available. However, this will not give you a good idea of how the UI will look on the mobile device, nor will it allow you to adequately test the functionality (such as tapping or swiping the mobile device screen). Therefore, it is important to test on the mobile device that you are developing for.

> **i** Note
>
> This URL addition is not valid for testing the screens with the **Mobile HTML** tag for the new Business ByDesign Mobile App

1. Develop your solution in the studio and make the work center view available to a mobile device.
2. In the SAP cloud solution, assign access rights to your business user so that they can access the new work center.
3. Download and install the SAP cloud solution on your mobile device.
4. On the mobile device, enter the URL of the development system.
5. Log on to the SAP cloud solution on your mobile device and test your solution.

9.6.2 Tasks

9.6.2.1 Modify an Object Work List for Use on a Mobile Device

Overview

You can modify an object work list created for use on a mobile device, using the user interface designer, in the studio.

Prerequisites

- You have created and activated the business object (BO) and other components in the studio.
  For more information, see Business Objects Quick Guide [page 398].
- You have created the object work list in the studio.
  For more information, see Generate Screens for a Business Object [page 429].

Procedure

1. In the studio, double-click the object work list in the *Solution Explorer*.

The object work list opens in the user interface designer.

2. In the *Properties Explorer*, under *Configuration-Tags* click *...*.
3. In the *Tags Editor* dialog box, do the following:
    1. Click *Add Operation*.
    2. In the new row that appears:
        - Under *Attributes*, select *Device*.
        - Under *Value*, select:
            - *Mobile* if you want to create screens for the old (orange) ByDesign Mobile App.
            - *Mobile HTML* if you want to create screens for the new Business ByDesign Mobile App released in November 2018.
    3. Click *OK*.
4. On the *Designer* tab, select the first column of the object work list.
5. In the *Properties Explorer*, do the following:
    1. Under *Appearance- DisplayType*, select **`Link`**.
    2. Under *Events- OnClickEvent*, select **`Edit Handler`**.
6. On the *Designer* tab, in the toolbar, delete all the buttons except the *New* button.

    > **i** Note
    >
    > The *New* button is linked to the + of the object work list on the mobile screen.

7. On the *Designer* tab, select the list.
8. In the *Properties Explorer*, under *ChildElements-List* columns click *...* .
9. In the *Columns Collection Editor* dialog box , select all columns except the first, click *Remove* and then, click *OK*.
10. In the dialog box that opens, click *Yes* to delete the data fields associated with the deleted columns.
11. In the *Properties Explorer*, under *Appearance-ShowHeaderRow* select **`False`**.
12. Save your changes.

9.6.2.2 Modify a Quick Activity for Use on a Mobile Device

Overview

You can modify a quick activity floorplan created for use on a mobile device, using the user interface designer, in the studio.

Prerequisites

- You have created and activated the business object (BO) and other components in the studio.
  For more information, see Business Objects Quick Guide [page 398].
- You have created the quick activity in the studio.
  For more information, see Generate Screens for a Business Object [page 429].

Modify a Quick Activity of a Business Object without Actions for Use on a Mobile Device

1. In the studio, double-click the quick activity floorplan in the *Solution Explorer*.
   The quick activity floorplan opens in the user interface designer.
2. In the *Properties Explorer*, under *Configuration-Tags* click *...*.
3. In the *Tags Editor* dialog box , do the following:
   1. Click *Add Operation*.
   2. In the new row that appears:
      - Under *Attributes*, select *Device*.
      - Under *Value*, select:
        - *Mobile* if you want to create screens for the old (orange) ByDesign Mobile App.
        - *Mobile HTML* if you want to create screens for the new Business ByDesign Mobile App released in November 2018.
   3. Click *OK*.
4. On the *Controller* tab, under *EventHandlers*, rename the *SaveHandler* to *SaveAndCloseHandler*.
5. Under *Configure the operation*, change *Action Type* to *SaveAndClose*.
6. Save your changes.

Modify a Quick Activity of a Business Object with Actions for Use on a Mobile Device

1. In the studio, double-click the quick activity floorplan in the *Solution Explorer*.
   The quick activity floorplan opens in the user interface designer.
2. In the *Properties Explorer*, under *Configuration-Tags* click *...*.
3. In the *Tags Editor* dialog box , do the following:
   1. Click *Add Operation*.
   2. In the new row that appears:
      - Under *Attributes*, select *Device*.
      - Under *Value*, select:
        - *Mobile* if you want to create screens for the old (orange) ByDesign Mobile App.
        - *Mobile HTML* if you want to create screens for the new Business ByDesign Mobile App released in November 2018.
   3. Click *OK*.
4. On the *Designer* tab, in the toolbar, delete all the buttons except the *Close* button.
5. Select the *Close* button, and in the *Properties Explorer* do the following:
   1. Under *Behavior-Enabled*, select **`False`**.
   2. Under *Behavior-Visible*, select **`False`**.
6. Select the contextual navigation region.
7. In the *Properties Explorer*, under *Toolbar-Configure* click *...*.
8. In the *Configure Toolbar* dialog box, do the following:
   1. In the *Available* list of buttons, under *ApplicationSpecific* buttons select *My Button*.
   2. Add *My Button* to the list of *Current* list of buttons using the right arrow button.
9. In the *Properties Explorer*, under *MenuInformation-HasMenu* select **`True`**.
10. In the *Properties Explorer*, under *MenuInformation-SubMenu* click *...* .
11. The *Button SubMenu Collection Editor* dialog box opens.
12. To add a new navigational item to the button sub menu, click *Add*.
13. For each action that you want to be displayed on the mobile device, add one navigational item.
14. Select the first navigational item and in the *Properties Explorer* do the following:
    1. Under *Events-OnClick* select *SaveHandler*.

2. Under *Design-Title* enter the title of the button as **Save**.

15. Select the second navigational item and in the *Properties Explorer* do the following:
    1. Under *Events-OnClick* select *Action<Name of Action in BO>Handler*.
    2. Under *Design-Title* enter the title of the button as **<Name of Action>**.
16. Repeat steps 15 and 16 for all other actions of the business object that you want to use.
17. Click *OK*.
18. In the *Properties Explorer*, under *Text Information-Text* enter the text **Actions**.
19. Save your changes.

9.6.2.3 Modify a Fact Sheet for Use on a Mobile Device

Overview

You can modify a fact sheet created for use on a mobile device using the user interface designer in the studio.

Prerequisites

- You have created and activated the business object (BO) and other components in the studio.
  For more information, see Business Objects Quick Guide [page 398].
- You have created a fact sheet in the studio.
  For more information, see Generate Screens for a Business Object [page 429].

Procedure

1. In the studio, double-click the fact sheet in the *Solution Explorer*.
   The fact sheet opens in the user interface designer.
2. In the *Properties Explorer*, under *Configuration-Tags* click ....
3. In the *Tags Editor* dialog box, do the following:
   1. Click *Add Operation*.
   2. In the new row that appears:
      - Under *Attributes*, select *Device*.
      - Under *Value*, select:
        - *Mobile* if you want to create screens for the old (orange) ByDesign Mobile App.
        - *Mobile HTML* if you want to create screens for the new Business ByDesign Mobile App released in November 2018.
   3. Click *OK*.
4. On the *Designer* tab, delete the *View All* button.
5. Select the *Close* button, and in the *Properties Explorer* do the following:
   1. Under *Behavior-Enabled*, select `False`.

2. Under *Behavior-Visible*, select **`False`**.

6. Select the fields present in the pane container of the fact sheet, and for each, in the *Properties Explorer*, under *Appearance-DisplayType* select **`Static text`**.
7. Delete the other pane containers in the fact sheet.
8. Select the section group and do the following:
    1. Double-click the section group name.
    2. In the *Dependent Property Editor* dialog box, clear the text used for *SectionGroup Name*.
    3. Click *OK*.
9. Select the section group and in the *Properties Explorer*, under *Text Information-Show Header* select **`False`**.
10. Delete the extended identification regions (Exldr) from the fact sheet.
11. Save your changes.

9.6.2.4 Configure a Fact Sheet for Use on a Mobile Device

Overview

You can configure a fact sheet for use on a mobile device. For example, you can configure a fact sheet so that you can navigate from an object work list to a fact sheet rather than navigating to a quick activity and then to a fact sheet .

Prerequisites

- You have created and activated the business object (BO) and other components in the studio.
  For more information, see Business Objects Quick Guide [page 398].
- You have created the following in the studio:
    - Work center view
    - Port type package
    - Object work list
    - Quick activity floorplan
    - fact sheet

  For more information, see Generate Screens for a Business Object [page 429].
- You have modified the fact sheet created for use on a mobile device, using the user interface designer, in the studio.
  For more information, see Modify a Fact Sheet for Use on a Mobile Device [page 781].

Process Flow

1. Assign the fact sheet to a work center view.
   For more information, see Assign a Mobile Fact Sheet to a Work Center View [page 783].

2. Create a port to allow the transfer of data between the fact sheet and other floorplans.
   For more information, see Create a Port to Navigate to a Mobile Fact Sheet [page 783].
3. Configure navigation in the mobile fact sheet to allow the transfer of data between the fact sheet and the object work list.
   For more information, see Configure Navigation in a Mobile Fact Sheet [page 784] and Configure Navigation in a Mobile Object Work List [page 784].
4. Configure an event handler in the mobile fact sheet.
   For more information, see Configure an Event Handler in a Mobile Fact Sheet [page 785].
5. Configure navigation between the mobile fact sheet and a mobile quick activity to allow the transfer of data from the quick activity to the fact sheet.
   For more information, see Configure Navigation between a Mobile Fact Sheet and a Mobile Quick Activity [page 786].

9.6.2.5 Assign a Mobile Fact Sheet to a Work Center View

Overview

You can assign a fact sheet that has been modified for use on a mobile device to a work center view.

Procedure

1. In the studio, double-click the work center view in the *Solution Explorer*.
   The work center view opens in the user interface designer.
2. In the *Properties Explorer*, under *RBAM Data-AssignedObjects* click *...* .
3. In the *Add Assigned Objects* dialog box, do the following:
   1. Click *Add* to add a new assigned object.
   2. In the new row, click *...* and select the fact sheet you have created from the list of repository items.
   3. Click *OK*.
4. Save and activate your changes.

9.6.2.6 Create a Port to Navigate to a Mobile Fact Sheet

You can create a new port to facilitate navigation to a fact sheet for use on a mobile device.

Procedure

1. In the studio, double-click the port type package in the *Solution Explorer*.
   The port type package opens in the UI designer.

2. Right-click the port type package and add a new port .
3. Rename the port as **`To_FS`**.
4. Save and activate your changes.

9.6.2.7 Configure Navigation in a Mobile Fact Sheet

Overview

You can configure navigation in a fact sheet to facilitate the transfer of data between the fact sheet and a object work list.

Procedure

1. In the studio, double-click the mobile fact sheet in the *Solution Explorer*.
   The fact sheet opens in the user interface designer.
2. On the *DataModel* tab, right-click on *Root* and add a new data field.
3. Rename the data field as **`Node ID`**.
4. On the *Controller* tab, right-click on *Inports* and add a new inport.
5. Under *Inport Configuration* do the following:
   1. Select the port type package that you have created.
   2. Set the port type reference to the port **`To_FS`**.
   3. Add a parameter and bind it to the data field **`Node ID`** that you have created.
6. Save and activate your changes.

9.6.2.8 Configure Navigation In a Mobile Object Work List

Overview

You can configure navigation in a object work list to facilitate data exchange to a fact sheet.

Procedure

1. In the studio, double-click the object work list in the *Solution Explorer*.
   The object work list opens in the user interface designer.
2. On the *Controller tab*, right-click on *Outports* and add a new outport.
3. Rename it as **`OutportToFS`**.

4. Under **Outport Configuration** do the following:
    1. Select the port type package that you have created.
    2. Set the port type reference to the port **`To_FS`**.
    3. Add a parameter and bind it to the data field **`Node ID`**.

    > **i** Note
    >
    > The parameter name should be the same in the fact sheet and the object work list.
5. On the *Designer* tab, select the column in the advanced list pane.
6. In the *Properties Explorer*, under *Appearance-DisplayType* select **`Link`**.
7. In the *Properties Explorer*, under *Navigation Information-Navigation* click *...*.
8. In the *Model Navigation* dialog box, do the following and click *OK*:
    1. Click on *Simple navigation*.
    2. Under *Navigation Details*, set the target as the fact sheet you have created.
    3. Under *Inport*, choose the inport you created in the fact sheet.
    4. Under *Outport*, choose the outport you created in the object work list.
    5. Select the parameters under *Outport Parameters* and *Inport Parameters* and click *Bind*.
9. Save and activate your changes.

9.6.2.9 Configure an Event Handler in a Mobile Fact Sheet

Overview

You can configure an event handler in a fact sheet to display data from a quick activity.

Procedure

1. In the studio, double-click the fact sheet in the *Solution Explorer*.
   The fact sheet opens in the user interface designer.
2. On the *Controller* tab, right-click on *EventHandlers* and add a new event handler.
3. Rename the event handler as **`ReadHandler`**.
4. For the **`ReadHandler`**, under *ReadHandler : Operations*, select the operation type **`BO Operation`**.
5. Set the *BO Operation* as **`Read`**.
6. Add a parameter, select **`nodeId`** as parameter type and bind it to the data field **`NodeId`**.
7. Bind the path of the parameter to the data field **`NodeId`**.
8. Select the inport you have created in the fact sheet.
9. In the *Properties Explorer*, under *Events-OnFire* select **Readhandler**.
10. Save and activate your changes.

9.6.2.10 Configure Navigation Between a Mobile Fact Sheet and a Mobile Quick Activity

Overview

You can configure navigation between a fact sheet and a quick activity so that it is possible to transfer data between them.

Procedure

1. In the studio, double-click the fact sheet in the *Solution Explorer*.
   The fact sheet opens in the user interface designer.
2. On the *Designer* tab, select the *Edit* button.
3. In the *Properties Explorer*, under *Misc-Name*, rename the button as **`EditEH`**.
4. On the *Controller* tab, right-click on *Outports* and add a new outport.
5. Under *Outport Configuration* do the following:
   1. Select the port type package that you have created.
   2. Set the port type reference to the port **`To_<Floorplanname>_edit`**.
   3. Add a parameter and bind it to the data field **`Node ID`**.
6. On the *Designer* tab, select the *Edit* button.
7. In the *Properties Explorer*, under *MenuInformation-Navigation* click *...*.
8. In the *Model Navigation* dialog box, do the following and click *OK*:
   1. Click on *Simple navigation*.
   2. Under *Navigation Details*, set the target as the quick activity you have created.
   3. Under *Inport*, choose the inport **`OBNEditInport`**.
   4. Under *Outport*, choose the outport you created in the fact sheet.
   5. Change *Style* to **`inplace`**.
   6. Select the parameters under *Outport Parameters* and *Inport Parameters* and click *Bind*.
9. On the *Controller* tab, rename the event handler generated by the simple navigation to **`EditEH`**.
10. Save and activate your changes.

9.6.2.11 Extend SAP Business ByDesign Mobile App Using SAP Cloud Applications Studio

With the **SAP Business ByDesign Mobile App**, you can access many of the main functions of the system that have been tailored to business on-the-run. Changes made on your mobile device are automatically updated in the system, over the Internet, online and in real time.

As in the desktop applications, you can use the ByDesign extensibility and flexibility features to extend the standard contents of the mobile app, as well as the content developed by partners and customers. In addition, you can use the SAP Cloud Applications Studio to extend many mobile functions.

This document describes the different extension possibilities and guides you to work with the basics for mobile extension development in the SAP Cloud Applications Studio.

Supported platforms

The mobile application is currently available for the following smartphones:

- Android Phones
- Apple iPhone

Extending the screens

You can extend the screens in two ways: either by extending the standard screens in ByD mobile app, or by creating new scenarios from scratch.

Extending existing standard screens in ByD Mobile App

For adding extensions to the existing mobile screens the same principles apply as for the desktop. Use the *Extensibility Explorer* of the Cloud Application Studio UI Designer and perform the required extensions. For more information, see Create a Business Object Extension [page 442].

Note: As on desktop, only those areas on a screen can be extended where stable anchors have been provided by standard development. To find out which screens can be extended on a mobile device, follow these steps:

1. Open the following URL on the Desktop Browser (e.g. Chrome) where <myTenantURL> is the URL of your tenant.
   https://<myTenantURL>/sap/public/ap/ui/repository/SAP_UI/HTMLOBERON5/client.html?sapbyd-agent=MBLHTML
2. Logon to the tenant.
3. Navigate to the screen for which you want to find out the path
4. Press key combination **Ctrl + Shift + Alt + Delete** for approximately 3 seconds. The technical information window opens
5. Click the *Controllers/Data* tab to find the path to the floorplan.

Creating a new scenario for the Business ByDesign Mobile App from scratch

The second approach is to develop new content by adding launchpad tiles, OWLs, factsheets, and other floorplan types. With this approach, you can create new business scenarios for the mobile app.

Create Desktop Screens

1. In the *Solution Explorer*, do one of the following:

- If you generate screens based on an SAP business object, proceed as follows:
  1. Right-click the project in your solution and select *Add* > *New Item*.
  2. Select the *Business Object* template, enter a name and click *Add*.
- If you generate screens based on one of your business objects, right-click the business object and select *Create Screens*.

2. In the *Create Screens* dialog box, enter a name for the new screen.
   The system uses this name together with the abbreviation of the screen type as a file name as follows: `[screen name]_OWL.OWL.uicomponent`.
3. Under *Supported Devices*, select *Desktop Only*.
4. Under *Screen Creation*, select one or more of the following screen types:
   - Object Instance Floorplan (OIF)
   - Quick Activity Floorplan (QAF)
   - Fact Sheet (FS)
   - Object Work List (OWL)
   - Object Value Selector (OVS)
   - Work Center View (WOCVIEW)

   For detailed information on the various screen types, see Screen Types [page 408].

   > **i Note**
   >
   > **Short ID**
   >
   > If you select *Work Center View* or a screen scenario containing a work center view, an additional field appears where you must enter a *Short ID* of up to 13 characters. It is preceded by the technical name of the solution and followed by **`_WCVIEW`**: `[Technical name of solution]_[Short ID]_WCVIEW`. The short ID helps you identify the work center view in the UI designer and in the SAP cloud solution. After you have entered the short ID, it is displayed in the following places:
   >
   > - UI designer: in the *Property Explorer* under *Component Info* in the *Short ID* field
   > - SAP cloud solution: *Application and User Management* > *Work Center and View Assignment*

5. Click *OK*.

Create Mobile Screens

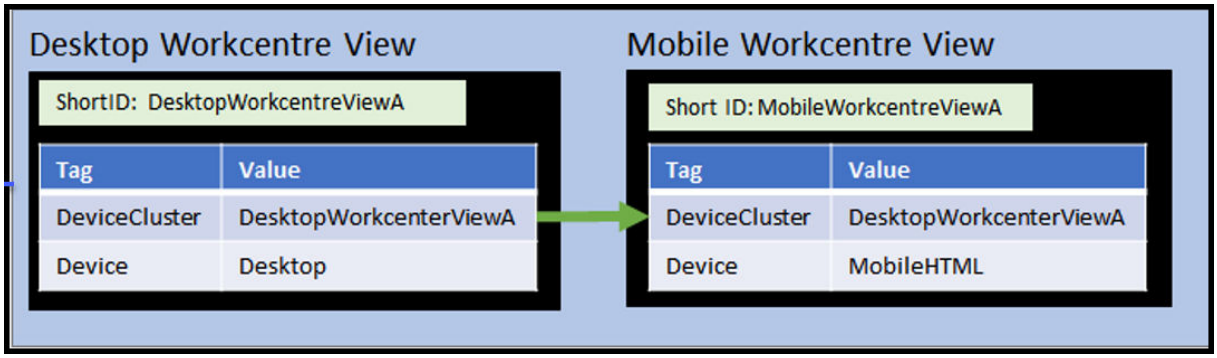
Follow the same steps as before with the following difference: Under *Supported Devices*, select *Mobile HTML*.

Set up Tags and Device Cluster

If you have more than one view of a work center (for example, a desktop view and a mobile view), you can use the *Device Cluster* tag in the UI designer to model the relationship between the views. The user assignment takes place on the leading work center view, which is always the desktop view. The view to render will be decided at runtime based on the user client.

The leading work center view (`WCV`) must have a *ViewSubType* of *Application*. You then create a *Device Cluster* and set the value to the *Short Id* of the `WCV`. To associate a mobile view, you give it a *ViewSubType* of *DeviceView* and assign it to the Desktop Workcentre View *Device Cluster*.

The other possible value for the *ViewSubType* is *InvisibleDeviceView*. You would select this if you do not want the view to be displayed on the UI, but want the view to serve as a carrier of authorizations only.



1. In the studio, double-click the object work list in the *Solution Explorer*.
   The object work list opens in the user interface designer.
2. In the *Properties* tab, under *Tags* click *...*.
3. In the *Tags Editor* dialog box, do the following:
   1. Click *Add Operation*.
   2. In the new row that appears:
      - Under *Attributes*, select *Device Cluster*.
      - Under *Value*, select the Short ID of the desktop floorplan you just created.
   3. Click *OK*.
4. Navigate to the *Solution Explorer*, and double-click the object work list for the mobile work center view to open the object work list in the UI designer.
5. In the *Properties* tab, under *Tags* click *...*.
6. In the *Tags Editor* dialog box, do the following:
   1. Click *Add Operation*.
   2. In the new row that appears:
      - Under *Attributes*, select *Device Cluster*.
      - Under *Mobile HTML*, the Short ID of the mobile floorplan you just created.
   3. Click *OK*.
7. Save your changes.

The connection between these two workcenter views are now established.

Supported Floorplans

Floorplans	Supported in Mobile Apps
Factsheets	Yes
Quick Activity Floorplans (QAF)	Yes Even though QAFs are supported by the mobile client, we recommend using OIFs instead. Multi-tab UI OIFs offer a better built-in scrolling behavior within tabs.
Object Instance Floorplans (OIF)	Yes
Object Worklists (OWL)	Yes
Embedded Components (EC)	Yes
Modal Dialogs (MD)	Yes

Floorplans	Supported in Mobile Apps
Action Forms	Yes
Guided Activity Floorplans	No

Assign Workcentre to User

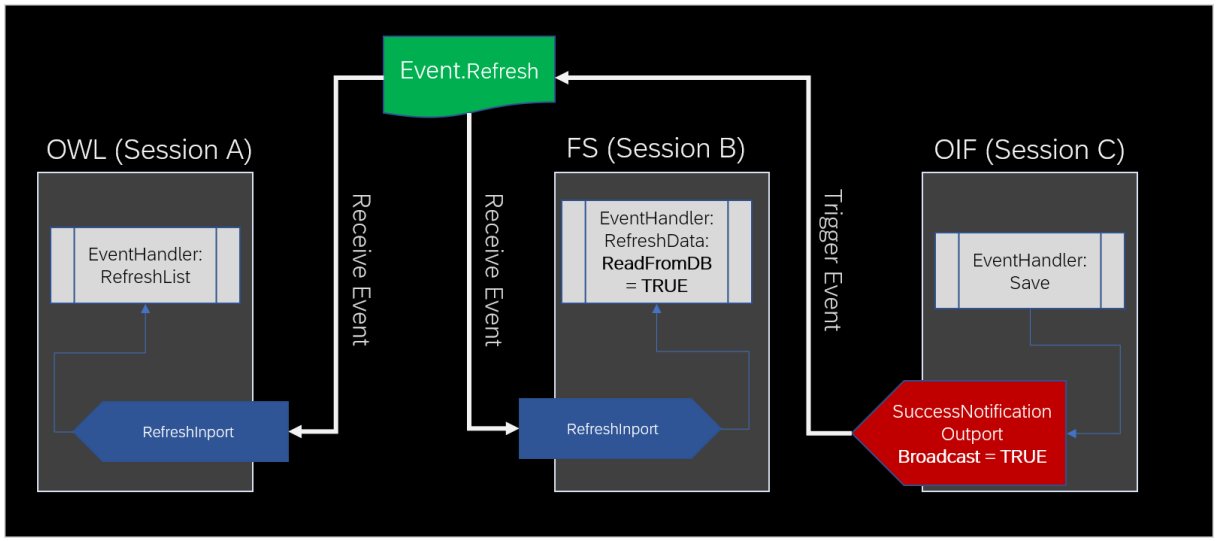
In the *Application and User Management* work center view, assign the workcenetre to the user. For more information, see Define Access Control [page 424]

Refresh Data of Stacked UIs

When you make changes in the OIF, the changes are not automatically reflected in the underlying factsheet and OWL. You must manually refresh it using *Success Notification Outports* > *Local Events*

We recommend that when you save the changes in the OIF, trigger the local event in the EventHandler that initiates the save.

After the event is triggered, all existing floorplans in the stack that have an inport modelled are notified. The inport is responsible for triggering the corresponding refresh-action depending on floorplan type and the mechanism by which the data is retrieved.



Triggering Screen

1. In the *Controller* tab, create an outport, and call it, for example, **LocalRefreshOutport**.
2. Define a unique name for the local event. We recommend that you follow a naming pattern to avoid duplication with other applications. For example **`<ApplicationArea>.<Object>.<Action>`** (e.g. MobileAccounts.Account.Updated)
3. In the *Properties* editor of the outport, set the *Broadcast* flag to *True* to ensure that the event is broadcast across all windows.
4. In the Event handler that contains the WindowAction for the *Save* operation, select **LocalRefreshOutport** for the Success Notification Port property. Whenever the save is triggered via this handler, the local event is triggered and broadcast to all open windows.

Receiving Screen

The receiving screens listen to the local events and trigger the corresponding refresh operation depending on the screen.

1. Create an inport, and call it, for example, **LocalEventRefresh**.
2. In the *Local Event* field enter exactly the same event name that you defined in the triggering screen.
3. In the *Properties* editor, select the eventhandler that reads / refreshs the data on the screen. This can be the Read-Handler that is loading the data when the screen is opened for the first time.

Session Agnostic Refreshs

There are cases where the stacked screens (OWL, FS, OIF) are open in different sessions. In such cases, the system might retrieve outdated data from the buffer and display wrong information on the UIs.

To avoid this, the *Read From DB* checkbox must be selected. You can find this property in the BO Operation for type *Read*. If set, the buffers are ignored and data is retrieved from the database.

10 Style Guides and Reference Guides

10.1 Overview of Style Guides and Reference Guides

Style Guides and Reference Guides encompasses the following:

Check Out the Style Guides	Check Out the Reference Guides
Style Guide for Writing System Message Texts [page 792] This style guide provides you with the standards and guidelines for writing system messages to be displayed on the user interface (UI) of your solution. It also provides the standards and guidelines for writing system messages that are relevant for SAP Support; these messages are not displayed on the UI. In addition, this style guide provides guidance on writing long-text documents providing additional information to a system message that helps the user to solve very complex error situations. These long-text documents can be incorporated into the Solution Documentation.	Reference Guide for Documentation of Cloud Solutions from SAP [page 821] This reference guide provides an overview of the main reference documentation types specific to the SAP cloud solution.

10.2 Style Guides

10.2.1 Style Guide for Writing System Message Texts

10.2.1.1 General Standards and Guidelines for Writing System Message Texts

Overview

This style guide provides you with the standards and guidelines for writing system messages to be displayed on the user interface (UI) of your solution. It also provides the standards and guidelines for writing system messages that are relevant for SAP Support; these are not displayed on the UI.

In addition, this style guide provides guidance on writing long-text documents providing additional information to a system message that helps the user to solve very complex error situations. These long-text documents can be incorporated into the Solution Documentation.

Target Group

This style guide is aimed at solution partners and reseller partners, who develop solutions in the studio for SAP cloud solutions.

Prerequisites

You are familiar with the following documents:

- User Interface Style Guide for Partners
- User Interface Style Guide
- Mobile Style Guide

Types of System Message Texts

System messages in an SAP cloud solution comprise the following message types:

- System message short texts
  - UI system message short texts
    System messages written directly for the UI address the end user directly. They provide concrete instructions on how to solve an error on the UI and contain terminology relevant to the SAP cloud solution.
  - Other system message short texts
    System messages written for developers and SAP Support. These messages are not aimed at the end user; they can be quite technical, describing the error situation and the reasons why an error occurred.
- Long-text documents with additional help to a UI system message short text.
  For very complex error situations, where the user needs to be provided with detailed background information about the error, a document can be written and provided as part of the solution documentation.

10.2.1.2 General Standards for Message Short Texts

Overview

This section of the style guide provides you with the general standards and guidelines for writing message short texts.

Standard formulations for different categories of message short texts are contained in Standard Formulations for Message Short Texts [page 797].

Content

1. Diagnose and describe the error.

   > **i Note**
   >
   > For technical reasons, system messages are restricted to 73 characters. If there is not enough space, for very complex error situations, where the user needs to be provided with detailed background information about the error, a long text can be written and provided as part of the solution documentation. For more information, see the section on General Standards and Guidelines for Message Long Texts [page 813].

2. If possible, provide a reason for the error.
3. If required, provide instructions on how to solve the error.

> **Example**
>
> - **Correct:** Currency missing
> - **Correct:** Creation not possible, source object does not exist

Sentence Construction

1. Formulate simple messages that are easily understandable.
2. Use precise and unequivocal language. Provide a maximum of information in a minimum of space.

   > **Example**
   >
   > - **Correct:** Recipient supply planning area not active
   > - **Incorrect:** Recipient supply planning area has wrong status
   >
   > **Note:** It is more precise to name the correct status.

3. Use sentence style.

   > **i Note**
   >
   > Some of the messages that appear in application logs deviate from this standard. They are used as headings (title case). For more information, see the section on Standards for Application Log Messages [page 808].

4. Do not use a period or exclamation mark at the end of a message short text.
5. If the message text consists of two parts, separate them using a semicolon (;)

   > **Example**
   >
   > - **Correct:** Error during RFQ creation; submission deadline missing
   > - **Incorrect:** Error During RFQ Creation! Submission Deadline Missing.

6. Do not use an article at the start of a message.

> **Example**
>
> - **Correct:** Hierarchy &1 already exists
> - **Incorrect:** The hierarchy &1 already exists

7. Omit articles within the text unless this creates ambiguity.

> **Example**
>
> - **Correct:** Buyer not authorized to display master data
> - **Incorrect:** Buyer not authorized to display the master data

8. Use the tense that is appropriate to the context.

> **Example**
>
> - **Correct:** Payment order creation failed
> - **Incorrect:** Payment order creation fails
> - **Incorrect:** Payment order creation will fail

Word Choice

1. Use terminology from the **Public Solution Model** for system messages that do not appear on the UI.

> **Example**
>
> - **Correct:** RFQ request currency missing
> - **Incorrect:** Sourcing request currency missing
>
> **Note:** The business object is named `RFQ Request`. On the UI the term used is *Sourcing Request*.

2. Use terminology from the SAP cloud solution for UI messages.

> **Example**
>
> - **Correct:** Sourcing request currency missing
> - **Incorrect:** RFQ request currency missing
>
> **Note:** The business object is named `RFQ Request`. On the UI the term used is *Sourcing Request*.

3. Avoid abbreviations wherever possible.
4. Do not use generic modeling terms such as **`type code`**.

> **Example**
>
> - **Correct:** Operation of type &1 not permitted
> - **Incorrect:** Operation category type code &1 not permitted

5. Do not use the word **`please`**.

> **Example**
>
> - **Correct:** Activate the document

- **Incorrect:** Please activate the document

Special Characters and Placeholders

1. Do not use special formatting, for example, upper case letters for emphasis. In some languages, the use of upper case can be construed as shouting.

   ✤ Example

   - **Correct:** Error during customer invoice creation
   - **Incorrect:** ERROR during customer invoice creation

2. Do not use quotation marks ("...") to set off placeholders, statuses, or similar.
3. Do not use special characters such as angle brackets (<...>), dollar signs ($), or asterisks (*).
4. Use ampersands (&) only for variables. If you use more than one variable, number your variables (&1, &2, &3, &4).
5. Do not write a message text that only consists of variables (for example, `&1 &2 &3`). Bear in mind that different languages have different word order. A message short text that reads correctly in the source language may be nonsensical in another language.
6. If a noun is preceded by a variable that is a placeholder for a cardinal number, do not set off the plural form or the plural ending of noun and verb with parentheses ((...)) or a slash (/). Use only the plural form.

   i Note

   Adding the plural endings in parentheses or with a slash does not work with irregular plural forms and with every language. Using the plural form may sometimes result in incorrect sentences, such as `1 documents not available`, but in the majority of cases the variable will be replaced with a number higher than `1`.

7. In general, avoid using formulations that cause problems in translation, especially in texts containing placeholders and ordinal numbers.

✤ Example

- **Correct:** Deletion of quantities not possible
- **Correct:** Line &1 deleted
  Note: This message is displayed on the UI as, for example, `Line 22 deleted`.
- **Correct:** Enter a valid quantity
- **Correct:** Enter a valid value in the &1 field
  Note: In this example, &1 stands for the Quantity field label (UI text). This message is displayed on the UI as `Enter a valid value in the Quantity field`.
- **Incorrect:** Cannot &1 quantities
  Note: Do not use a placeholder for a verb.
- **Incorrect:** &1th line deleted
  Note: Do not use a placeholder for an ordinal number. This message is displayed on the UI as `22th line deleted`, which is incorrect in English.
- **Incorrect:** Enter a valid &1

> **Note:** In this example, &1 stands for the Quantity field label (UI text). This message is displayed on the UI as `Enter a valid Quantity`. The capitalization is incorrect in English, and the adjective cannot be translated correctly into another language.

10.2.1.3 Standard Formulations for Message Short Texts

Overview

This section of the Style Guide and its subsections provides standard formulations for the most common message short texts. They are listed and structured according to the error category to which they belong. Whenever you write a message, analyze the error situation and decide to which category it belongs. If none of the categories listed here fits the error situation of your message, follow the General Standards for Message Short Texts [page 793] to write the message.

Error Categories

Errors and Failures	• Failure [page 798] • Errors [page 798] • Errors Due to Business Configuration [page 799]
Inability to Perform Action, Status and Authorization Problems	• Inability to Perform Action [page 799] • Inability to Perform Action Because Object is Locked [page 800] • Inability to Display Data [page 800] • Lack of Authorization [page 800] • Inappropriate Status [page 801] • Mandatory Step [page 801]
Existence and Availability of Objects, Missing Data, and Search Results	• Existence of Objects [page 802] • Multiple Existence of Objects [page 802] • Non-Existence of Objects [page 802] • Availability of Objects [page 803] • Missing Data [page 803] • Search Results [page 803] • Determination Problems [page 804]
Invalid Data and Value Problems	• Invalid Data [page 804] • Value Problems [page 804] • Assignment Problems [page 805] • Role Problems [page 805] • Date Problems [page 806]

Comparison Problems	• Missing Counterpart [page 806] • Difference in a Comparison [page 807]
Object Selection	Selecting Objects [page 807]

10.2.1.4 Standards for Errors and Failures

10.2.1.4.1 Failure — Standard Formulations

You use this standard formulation in cases where the execution of an action fails completely or where a general failure occurs.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
... failed; <instruction or reason>	Printing failed; install printer	Error during printing
	Printing failed; printer settings not correct	Printing had a failure
		Internal error; internal data inconsistency

> **Note**
>
> Printing failed means that a document has not been printed at all.

10.2.1.4.2 Errors — Standard Formulations

You use these standard formulations in cases where the execution of an action has not failed completely, but can only be completed with errors.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
Error while <operation>; <instruction or reason>	Error while processing incoming e-mail; check e-mail settings	Errors were found
Error during <operation>; <instruction or reason>	Error during RFQ creation; RFQ ID missing	Error occurred during RFQ creation

> **Note**
>
> Error while printing means that a document has been printed, but some pages are missing, for example.

10.2.1.4.3 Errors Due to Business Configuration— Standard Formulations

You use these standard formulations in cases where an error occurs due to wrong or missing business configuration.

> **i** Note
>
> If possible, provide detailed information about what is wrong in the business configuration in a message long text. For more information, see General Standards and Guidelines for Message Long Texts [page 813].

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
Error in business configuration for ...	Error in business configuration for purchasing contract	Business configuration error: Maintain purchasing contract
	Error in business configuration for pricing procedure **Note:** For the message long text, an example is `The system could not determine or assign the Catalog for Pricing field. ...`	Pricing not maintained in business configuration

10.2.1.5 Standards for Inability to Perform Action, Status and Authorization Problems

10.2.1.5.1 Inability to Perform Action — Standard Formulations

You use these standard formulations in cases where an action cannot be performed or is not permitted.

If the action is not followed by an object, use the technical action name, for example, `Delete`, `Activate`, and so on).

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Action name> not permitted; <instruction or reason>	Delete not permitted for user status; user not authorized to delete objects	Function prohibited for user status
<Action name> not possible; <instruction or reason>	Create not possible; source object does not exist	Creation not possible, because source object does not exist

If the action is followed by an object, you may use the gerund or another verbal noun, for example, **`Deleting the document`**, and so on.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
... <object> not permitted; <instruction or reason>	Deletion of quantities not permitted; user not authorized to delete quantities	Delete quantities not allowed; user not authorized to delete quantities
... <object> not possible; <instruction or reason>	Creating the document not possible; source object does not exist	Create document not possible; source object does not exist

10.2.1.5.2 Inability to Perform Action Because Object is Locked — Standard Formulations

You use these standard formulations in cases where an action cannot be performed because the object is being used by another user..

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
Operation not possible; <object> locked by <user>	Operation not possible; hierarchy locked by user &1	The operation cannot be performed because the hierarchy is locked by user &1
	Operation not possible; production order locked by user &1	Production order is locked by user &1

10.2.1.5.3 Inability to Display Data — Standard Formulations

You use this standard formulation in cases where the system cannot provide certain data to be displayed.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
Cannot display <object>; <instruction or reason>	Cannot display revision status; period already completed	You cannot display the revision status here

10.2.1.5.4 Lack of Authorization — Standard Formulations

You use this standard formulation in cases where a party or a user does not have the authorization to perform a certain step.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
... not authorized to ...	Buyer not authorized to display master data	No authorization to display master data
	User &1 not authorized to delete purchase order	The user does not have change authorization to delete the purchase order

10.2.1.5.5 Inappropriate Status — Standard Formulations

You use these standard formulations in cases where an object does not have the appropriate status. As a consequence, the user cannot change the object or use it in a transaction, for example.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Object> not <status>	Business partner &1 not active	Business partner &1 has not been activated
<Object> <status>	Bidder &1 not released	Release of bidder &1 missing. **Note:** The correct status name is **`not released`**.
	Delivery request inconsistent	Delivery request is not consistent **Note:** The correct status name is **`inconsistent`**.

10.2.1.5.6 Mandatory Step — Standard Formulations

You use this standard formulation in cases where a certain step must be performed.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
... required for ...	Mandatory step &1 required for phase &2	Perform step &1 before you proceed with phase &2
		Mandatory step &1 for phase &2 missing

10.2.1.6 Standards for Existence and Availability of Objects, Missing Data, and Search Results

10.2.1.6.1 Existence of Objects— Standard Formulations

You use this standard formulation in cases where an object already exists in the system.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Object> already exists	Hierarchy &1 already exists	There is already a hierarchy &1
	Open customer quote &1 already exists for buyer &2	An open customer quote &1 exists for buyer &2

10.2.1.6.2 Multiple Existence of Objects — Standard Formulations

You use this standard formulation in cases where an object exists in the system more than once, which means it is not unique.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Object> exists more than once	Party &1 exists more than once	Party &1 is not unique
		There is more than one party &1

10.2.1.6.3 Non-Existence of Objects — Standard Formulations

You use this standard formulation in cases where an object does not exist at all in the system, which means it has never been created or it has been deleted.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Object> does not exist	Number range &1 does not exist **Note:** Number range does not exist means that the number range has never been created.	Number range &1 not found **Note:** This message is only correct if it is displayed after a user-initiated search.

10.2.1.6.4 Availability of Objects — Standard Formulations

You use this standard formulation in cases where an object exists in the system but is not available for a certain action.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Object> not available	Seller &1 not available for purchasing unit &2 Note: `Seller &1 not available` means that the seller exists in the system, but it is not available for a certain action, for example, because it has not been assigned for that action.	Seller &1 is not defined for purchasing unit &2

10.2.1.6.5 Missing Data — Standard Formulations

You use this standard formulation in cases where a mandatory field has not been filled or a required value has not been transferred in an XML message.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Object missing>	Currency missing	Currency has not been defined
	Unit of measure for the quantity in item &1 missing	Enter a unit of measure for the quantity in item &1

10.2.1.6.6 Search Results — Standard Formulations

You use these standard formulations in cases where, during a user-initiated search, a specific object was found, could not be found, or no object was found at all.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Object> found	&1 entries found	&1 entries determined
<Object> not found	Specified material &1 not found	Specified material &1 could not be found
No <object> found	No documents found	No suitable entry was found

10.2.1.6.7 Determination Problem — Standard Formulations

You use this standard formulation in cases where the system cannot determine a certain object or value.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Object> not determined; <instruction or reason>	Product type for product &1 not determined; add product type to product	The system could not determine the product type for product &1
	Buying organizational unit not determined	Buying organizational unit not found **Note:** This message would only be correct if it was displayed after a user-initiated search

10.2.1.7 Standards for Invalid Data and Value Problems

10.2.1.7.1 Invalid Data — Standard Formulations

You use these standard formulations in cases where a certain value is not valid in the current context or if a certain value is not valid at all..

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
... not valid	Inventory valuation level &1 not valid	Inventory valuation level &1 is invalid
... not valid for ...	Number &1 not valid for external interval &2	Number &1 is invalid for external interval &2

10.2.1.7.2 Value Problems — Standard Formulations

You use these standard formulations in cases where a certain value is not permitted or wrong in a specific context.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Value> not permitted	Unit of measure &1 not permitted **Note:** &1 is a valid unit of measure, but it is not permitted in the context of the error message.	Unit of measure &1 not allowed
		Unit of measure &1 not valid
Only one <object> permitted	Only one buyer permitted	Enter exactly one party of role buyer

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
	Only one quantity permitted as valuation quantity	Only one quantity can be the valuation quantity
<Value> greater than <value>	Item value greater than limit &1 of purchase order &2 **Note:** This indicates that the item value is greater than allowed.	Item value is higher than limit &1 of purchase order &2 Item value exceeds limit &1 of purchase order &2
<Value> less than <value>	Purchase order net amount less than minimum order value **Note:** This indicates that the item value is less than allowed	Purchase order net amount is lower than minimum order value
<Value> must not be greater than <value>	Invoiced quantity must not be greater than delivered quantity for item &1 **Note:** This indicates that the invoiced quantity must be smaller than or equal to the delivered quantity.	Invoiced quantity must be less than or equal to the delivered quantity for item &1

10.2.1.7.3 Assignment Problems — Standard Formulations

You use this standard formulation in cases where a user or business partner has not been assigned.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
... not assigned to ...	Contact person &1 not assigned to bidder &2	Assignment missing for contact person

10.2.1.7.4 Role Problems — Standard Formulations

You use this standard formulation in cases where a user or business partner does not have the appropriate role.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
... does not have ... role; <instruction or reason>	&1 with ID &2 does not have &3 role; role must be assigned	Bus partner role cat &3 not maintained in master data for party &1 ID &2

10.2.1.7.5 Date Problems — Standard Formulations

You use these standard formulations in cases where there are problems with a date used in a business transaction.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<... date> must be before <... date>	Start date must be before end date	Enter a start date that is before the end date
<... date> must be after <... date>	Submission deadline must be after end date	Enter a submission deadline that is after the start date
<... date> must be the same as <... date>	Posting date must be the same as run date	Enter a posting date that is the same as the run date
<... date> must not be after <... date>	Source of supply date must not be after supply quota arrangement date **Note:** This indicates that the source of supply date must be before or the same as the supply quota arrangement date.	Date/Time From of SourceOfSupply is later than that of SQA
<... date> must not be in the future	Receipt date must not be in the future	Enter a receipt date that is not in the future
<... date> must not be before <... date>	Source of supply date must not be before supply quota arrangement date **Note:** This indicates that the source of supply date must be after or the same as the supply quota arrangement date.	Date/Time To of SourceOfSupply is earlier than that of SQA
<... date> must not be in the past	Start date must not be in the past	Enter a start date that is not in the past
<... period> must not start before <... date>	Binding period must not start before submission deadline **Note:** This indicates that the binding period must start on or after the date of the submission deadline	Enter a binding period that starts after the submission deadline

10.2.1.8 Standards for Comparison Problems

10.2.1.8.1 Missing Counterpart — Standard Formulations

You use this standard formulation in cases where the counterpart of a certain object cannot be found for a comparison.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
No ... matches ...	No accounting notification matches payment allocation &1	No accounting notification coincides with payment allocation &1
	No accounting notification matches supplier invoice	The corresponding accounting notification does not exist for supplier invoice

10.2.1.8.2 Difference in a Comparison — Standard Formulations

You use these standard formulations in cases where the following applies:

- Two values in a comparison are not the same
- One value in a comparison is not the same as the other
- The type or category of an object in a comparison does not match the type or category of another object

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
... differ	Transaction dates differ	Transaction dates mismatch
		Transaction dates do not coincide
... differs from ...	Currency &1 in confirmation differs from currency &2 in purchase order	Currency &1 in confirmation mismatches currency &2 in purchase order
	Price of item &1 in confirmation differs from that in purchase order	Price difference between confirmation and purchase order in item &1
... and ... do not match	Identified stock &1 and material &2 do not match	Identified stock &1 and material &2 differ

10.2.1.9 Standards for Object Selection

10.2.1.9.1 Selecting Objects — Standard Formulations

You use these standard formulations for error situations where the end user has to select an object.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
Select <object>	Select the document to be deleted	Choose the document to be deleted

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
Additional <objects> not selected	Additional status profiles not selected	Additional status profiles not highlighted

10.2.1.10 Standards for Application Log Messages

10.2.1.10.1 Standards and Guidelines for Application Log Messages

Overview

This section of the style guide and its subsections provides standards and guidelines for application log messages.

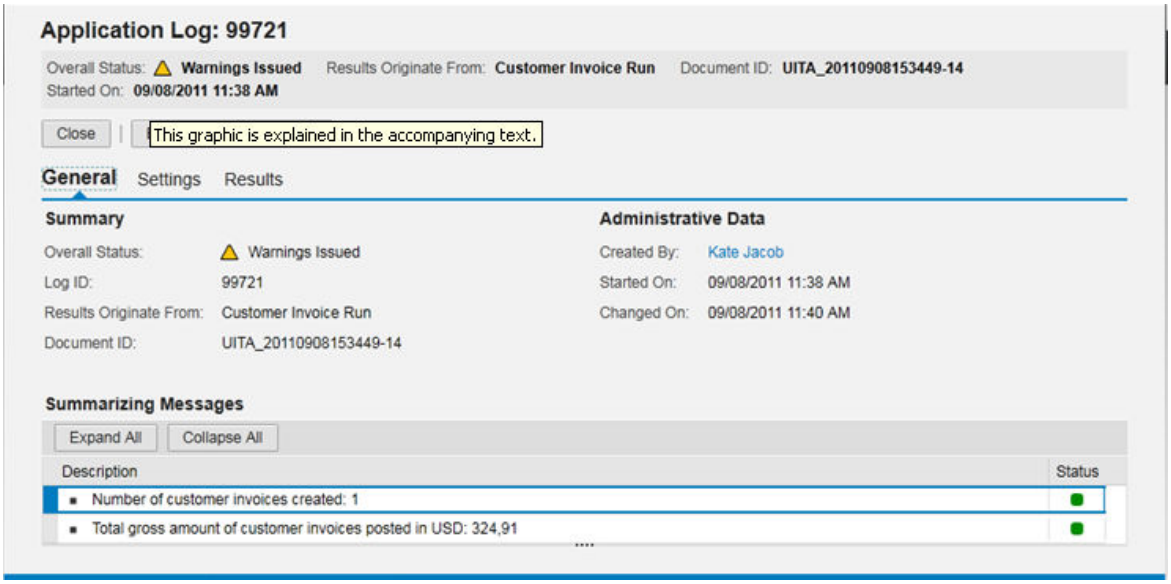
Application logs belong to mass data run objects (MDROs) and support mass data runs. When a run is started, it takes place in the background. At the end of the run, the application log is displayed on the user interface (UI) to retrace the results.

Most of the message texts that are used in application logs are not really error messages. They are used to provide texts for the different sections of the application log, such as headings or selection data.

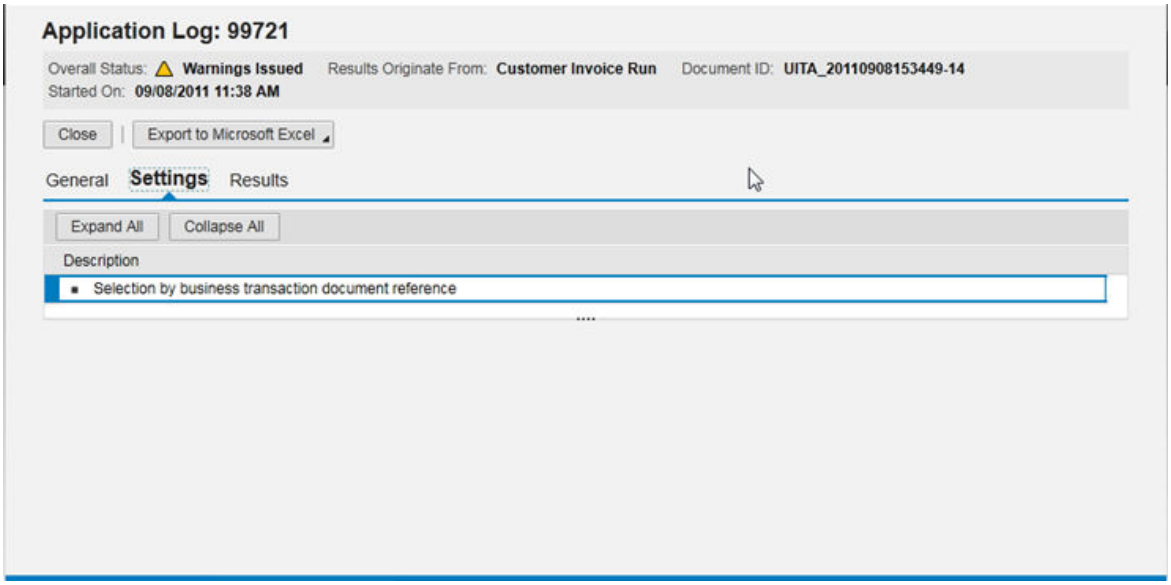
Structure of the Application Log

Application logs consist of three sections that group the messages posted:

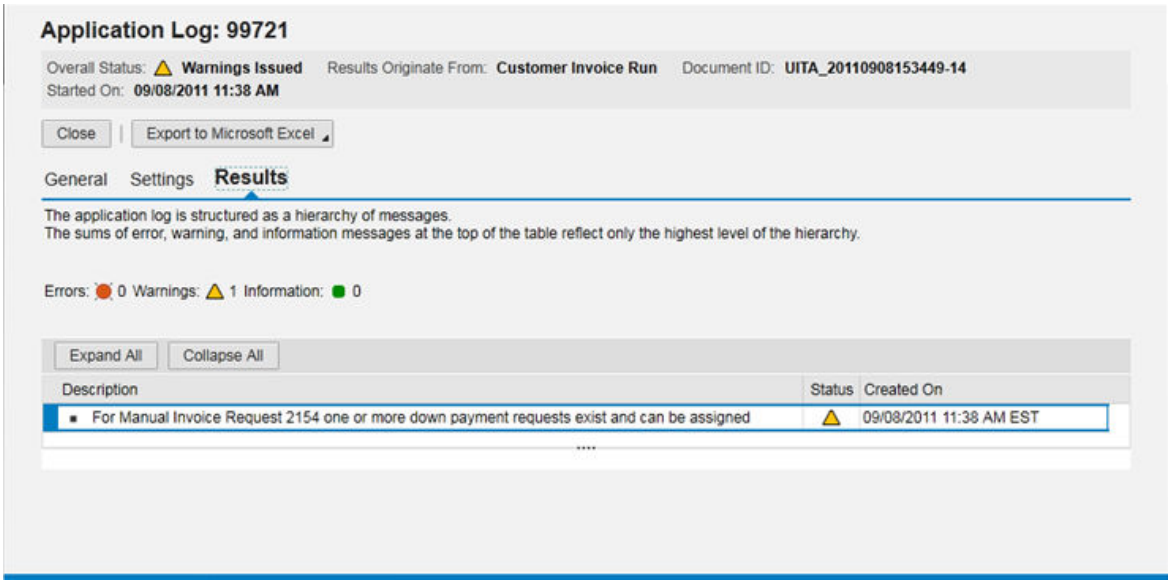
1. General
   This section contains a summary of what happened during the mass data run and generic log statistics, for example, the number of errors.



2. Settings
   This section contains information on the parameters and settings that have been selected for the mass data run.



3. Results
   This section contains detailed information about the decisions taken and about the errors and warnings that occurred during the mass data run in chronological order.



10.2.1.10.2 Number of Objects — Standards for Application Log Messages

You use these standard formulations for messages used in application logs to display the number of objects that were or were not processed correctly during an application log run.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
Number of ...: <number>	Number of procurement planning orders changed: &1	&1 procurement planning orders have been changed
Number of ... not ...: <number>	Number of procurement planning orders not changed: &1	&1 procurement planning orders were not changed
	Number of production planning orders not changed: &1	&1 production planning orders could not be changed

10.2.1.10.3 Headings — Standards for Application Log Messages

1. For messages used in application logs as headings, use title case.
2. Do not put a colon (:); semi-colon (;), or period (.) at the end of the message.

Example of Correct Usage	Example of Incorrect Usage
Bank Statement &1 **Note:** Under this heading, messages concerning bank statement &1 are listed	Bank Statement
Checking Parameters **Note:** Under this heading, all relevant checking parameters are listed	Checking parameters
Selection Data	Selection criteria:
Summary	Summary Details:

10.2.1.10.4 Selection By — Standards for Application Log Messages

You use this standard formulation to specify the selection category that defines the elements to be included in or excluded from an application log run.

You use title case to formulate the message.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Name of SelectionBy node with spaces>	Selection by Purchase Order	Selection by purchase order
	Selection by Buyer	Selection by BuyerParty

> **i** Note
>
> If the node name includes the word `party`, leave it out in the message.

10.2.1.10.5 Inclusion/Exclusion — Standards for Application Log Messages

You use these standard formulations for messages used in application logs to specify the elements excluded from or included in an application log run.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
Exclude all matches (with) <lower boundary type code>; <lower boundary ID>	Exclude all matches with &1 &2	Exclusion code: &

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
Include all matches from <lower boundary> to <lower boundary>	Include all matches from &1 to &2	Inclusion code: &1 &2

10.2.1.10.6 Control Parameters— Standards for Application Log Messages

You use this standard formulation for messages used in application logs to display the settings that are used for a control parameter.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
<Name of parameter element>: <Value of parameter element>	Employee time valuation period closure date: &1	The employee time valuation period closure date is &1

10.2.1.10.7 Application Log Run Start/End — Standards for Application Log Messages

You use these standard formulations for messages used in application logs to specify when an application log run has started or ended.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
... started	Adjustment run started	Adjustment run has been started
... ended	Cleanup run ended	Cleanup run has finished

10.2.1.10.8 Success or Failure Messages — Standards for Application Log Messages

1. For messages that describe the result of an application log run, use sentence style.
2. Do not use the words `successfully` and `unsuccessfully`.
3. Do not put a period at the end of the message text

Example of Correct Usage	Example of Incorrect Usage
&1 new period closure dates not created	& new period closure dates created unsuccessfully

Example of Correct Usage	Example of Incorrect Usage
&1 of &2 orders created	&1 of &2 order have been created successfully.

10.2.1.10.9 ID Messages — Standards for Application Log Messages

You use these standard formulations for messages used in application logs to display the ID of an object that has been processed during an application log run.

Standard Formulation	Example of Correct Usage	Example of Incorrect Usage
... ID: <ID>	Business transaction document ID: &1	The business transaction document ID is &1
	Customer requirement ID: &1	Customer Requirement ID: &1

10.2.1.11 General Standards for Message Long Texts

10.2.1.11.1 General Standards and Guidelines for Message Long Texts

Overview

This section of the Style Guide provides you with the general standards and guidelines for writing message long texts.

General

1. Add a section to the Solution Documentation for long texts that contain additional information on complex system messages. These are messages in which you cannot explain within 73 characters what the user has to do to resolve an error.
2. Name the section in the Solution Documentation **`Additional Information on System Messages`**.
3. Use the structure below for each message long text.
4. Separate the different message long texts by sub-sections.

Structure of System Message Long Texts

Keyblock: Title

1. This keyblock is mandatory. It links the long text to the short text in the system.
2. For the title, use the text of the message short text.
3. Do not use variables in the document title. If the system message contains variables, rephrase it to the document title.

Example of Correct Usage	Example of Incorrect Usage
Invoice may be a duplicate of another invoice	Invoice may be a duplicate of invoice &1

Keyblock: What Happened?

1. This keyblock is optional. It advises the user what caused the error and what implications the error has.
2. Answer the following questions:
    1. What is the cause of the error message?
    2. In which context does the error occur?
    3. What are the implications?

Keyblock: What Can I Do?

1. This keyblock is mandatory. It advises the user what they can do to resolve the error message.
2. Answer the question: What can the user do to solve the error?
3. If appropriate, use a numbered list.

Example

Message Short Text/Keyblock	Sample Text
Message Short Text	Invoice may be a duplicate of invoice &1
Message Long Text	
Keyblock: Title	Invoice may be a duplicate of another invoice
Keyblock: What Happened?	The system was unable to post the supplier invoice or credit memo and created an exception.
Keyblock: What Can I Do?	Check your entry and proceed as follows: 1. Open the *Supplier Invoice* editor and go to the *Exceptions* tab. Before resolving the exception, if you require further information from another person, click *Forward for Clarification*. 2. Compare the invoice details with the similar invoice already posted in the system. 1. If the invoice is not a duplicate, under *Exception Details*, click *Treat as New Invoice*. 2. If the invoice is a duplicate, under *Exception Details*, click *Treat as Duplicate Invoice*.

10.2.1.11.2 Standards for Message Long Texts for PAF BTM Tasks

Overview

Process Agent Framework Business Task Management (PAF BTM) tasks are used to inform a user about a manual cleanup activity that has to be done to solve a process communication error that occurred in inbound error conflict handling.

Each PAF BTM task has a task description, which provides detailed information on a fatal error that occurred and instructions on how to solve the error.

> ### Example
>
> The PAF BTM task description is structured as follows:
>
> A fatal error occurred:
>
> <PAF BTM Error Message Short Text>
>
> <PAF BTM Error Message Long Text>

The message short text is used to briefly describe the fatal error that occurred. For this text (called PAF BTM error message short text in this document), the General Standards for Message Short Texts [page 793] and the Standard Formulations for Message Short Texts [page 797] apply.

The message long text provides detailed information and instructions on how to solve the error. For this text (called PAF BTM error message long text in this document), follow the standards and guidelines listed below.

General Standards

1. Include these message long texts in the section of the Solution Documentation named **`Additional Information on System Messages`**.
   Create this section if it does not already exist.
2. Formulate the message long text according to the keyblocks described below.
3. Use the SAP cloud solution terminology.
4. If you want to use a variable, use the following format: &v1&. Do not use the variable format that is usually used for message short texts (&1).

Example of Correct Usage	Example of Incorrect Usage
Open the &v1& project	Open the &1 project.

5. Do not enter more than four variables.
6. Do not use abbreviations.

Keyblock: Context Information

1. This keyblock is optional.
2. Answer the following questions:
    1. What is the cause of the error message?
    2. In which context does the error occur?
    3. What are the implications?
3. Only use this keyblock if the message short text does not provide enough context information on the error.

> ### Example
>
> The status of the &v1& project and the &v2& task is On Hold, Stopped, or Closed. Therefore, no further postings are permitted. As a result, a time confirmation related to the &v1& project and the &v2& task could not be processed

Keyblock: Solution

General Standards

1. This keyblock is mandatory.
2. Answer the question: What can the user do to solve the error?
3. Use one of the following standard formulations:
    - If there is only one solution, use the following:
      `To solve the error, proceed as follows:`
    - If there is more than one solution, use the following:
      `You have the following options to solve the error:`
4. Describe the solution using a numbered list.
   Use the standard formulations in the sub-section below, If the message refers to one of the following:
    - Errors that refer to handling lock issues
    - Errors that occur when an inbound XML message that has been sent via a third party interface cannot be processed
5. In the second last step, describe how the system responds to the user action.
6. In the last step, describe what the user needs do on the *Process Communication Error* screen to finalize the task. The user has the following options:
    - Restart process
    - Reject process
    - Synchronize process
    - Mass restart process
    - Abort process

   For more information, see the *Keyblock: Result* section below.
7. If there is more than one solution to solve the error, describe all of them and introduce each of them with the following standard formulation:
   `Solution <number of solution>:`
   `Use this solution if <describe why this solution should be used> ...`

Example

- **Example of Only One Solution:**
  To solve the error, proceed as follows:
  1. Click *Open Business Document* to navigate to the &v1& project.
  2. Go to edit mode and navigate to the &v2& task.
  3. Change the task as follows:
     - If the project status is Stopped or Closed, change it to Released.
     - If the On Hold flag is selected for the task, deselect the flag.
  4. Save the changes.
  5. The system sets the project status of the &v1& project and the &v2& task to Released, and On Hold is no longer selected.
  6. Restart the process by clicking Restart Process on the Process Communication Error screen.
- **Example of More than One Solution:**
  You have the following options to solve the error:
  Solution 1:
  Use this solution if the quantity sent by the service confirmation appears to be incorrect.
  1. Clarify with the responsible person for the service confirmation, whether incorrect data was entered during service confirmation.
  2. Ensure that the responsible person for the service confirmation corrects the data in the service confirmation.
  3. The system reprocesses the data, which is now correct.
  4. Synchronize the documents by clicking Synchronize Documents on the Process Communication Error screen.

  Result: The system synchronizes the documents using the updated data of the service confirmation.
  Solution 2:
  Use this solution if the stock update given for the outbound delivery is incorrect.
  1. Find the corresponding outbound delivery.
  2. Make adjustment confirmation on the outbound delivery.
  3. The system accepts the reported quantity of the service confirmation.
  4. Restart the process by clicking Restart Process on the Process Communication Error screen.

  Result: The system updates the stock accordingly.

  > **Note**
  >
  > The message short text that belongs to this long text is “Negative stock not permitted in custodian party”. This task is created if the quantity reported in a service confirmation exceeds the allocated quantity of the material in the identified stock.

Standard Formulation for Handling Lock Issues

For message long texts that refer to handling lock issues, use the following standard formulations:

Keyblock: Solution and Step: Result	To solve the error, proceed as follows: 1. Open the Process Communication Error screen. 2. In the Related Errors list, identify the user locking the business document. 3. Ensure that the user closes the business document. 4. The system unlocks the document. 5. Restart the process by clicking Restart Process. Result: The system updates the business document

> **i Note**
>
> The standard formulation of short texts for messages referring to handling lock issues is one of the following:
>
> - Operation not possible; <business document locked>
> - <Operation> not possible, <business document> <business document ID> locked by <user name>

Standard Formulation for Inbound Third Party Communication Errors

For message long texts that refer to errors that occur when an inbound XML message that has been sent via a third party interface cannot be processed, use the following standard formulation:

Keyblock: Solution and Step: Result	You have the following options to solve the error: Solution 1: Use this solution if a business document that you have received via a third party interface has to be changed by the sender of the document. 1. Check the details of the error message on the Process Communication Error screen. 2. Contact the person responsible on sender side of the message to inform him or her about the issue: Party: &v1& Contact Person: &V2& Phone Number: &v3& Sender System: &v4& 3. Ask the person on sender side to perform the relevant changes (so that the message can be processed in your system) and to send the business document again. 4. Abort the process by clicking Abort Process on the Process Communication Error screen. 5. Restart the process by clicking Restart Process. Result: The system does not process the message, and the sender needs to resend a corrected message. Solution 2: Use this solution if you have to make changes to your system to enable the processing of a business document that you have received via a third party interface. 1. Check the details of the error message on the Process Communication Error screen. 2. Adjust the data in your system, for example, master data or business configuration data, or edit the payload, if allowed, on the Process Communication Error screen as required. 3. Restart the process by clicking Restart Process. Result: The system updates the business documents. **i Note** For messages referring to errors that occur when an XML message that has been sent via a third party interface could not be processed, he standard formulation of the message short text is: `Error while processing inbound third party communication`

Step: Result

1. This step is mandatory.
2. Describe the result of the steps performed to solve the error:

3. Depending on the user's options, use one of the following standard formulations:

User Option	Standard Formulation
Restart Process	Result: The system updates <object, including &v1&>. **i Note** ○ The standard formulation of the Result step for messages referring to handling lock issues is: Result: The system updates the business document ○ For messages referring to errors that occur when an inbound XML message that has been sent via a third party interface cannot be processed, the standard formulation of the Result step is: ○ For Solution 1: Result: The system does not process the message, and the sender needs to send a corrected message. Restart the process by clicking Restart Process. ○ For Solution 2: Result: The system updates the business documents.
Reject Process	Result: The system does not update <object, including &v1&>, and cancels the process communication.
Synchronize Documents	Result: The system synchronizes the documents using the updated data of the <object, including &v1&>.
Mass Restart Process	Result: The system updates <object, including &v1&> by restarting all related process communication.
Abort Process	Result: The system does not update <object, including &v1&>, and the process communication is aborted.

Example

Result: The system updates the &v1& project and the &v2& task with the time confirmation.

4. If there is more than one solution to solve the error, do one of the following:
   - If the result of each is different, add a Results keyblock after each solution.
   - If the result of all of them is always the same, only add one Results keyblock (after the last solution).

Keyblock: Further Information

1. This keyblock is optional.
2. If there are any follow-up activities required that are not directly related to the error solution, for example, to avoid data inconsistencies, describe them here.
3. Do not describe the follow-up activities in detail.

4. Do not use a numbered list.
5. Use the following standard formulation:
   `Note: <follow-up activity> ...`

> Example
>
> **Note:** Check if it should be permitted to add further postings to the &v1& project and the &v2& task:
>
> - If further postings are to be permitted, no further actions are required
> - If further postings are not to be permitted, change the task data back to the previous project status or set it to On Hold

10.3 Reference Guides

10.3.1 Reference Guide for Documentation for Cloud Solutions from SAP

10.3.1.1 Introduction to Documentation for Cloud Solutions from SAP

This reference guide provides an overview of the main reference documentation types specific to SAP cloud solutions. It covers the following:

- Business Configuration Documentation [page 826]
  - Business Adaptation Catalog
  - Activity List
  - Basic Data
- Work Center Documentation [page 830]
  - Analytics
    - Data sources: multidimensional analytical views (MDAVs), key figure structures, and key figure grids
    - Reports
    - Key performance indicators
    - Dashboards
  - Business Task Management
  - Business Background Documentation
    - Concept Documents
    - Process Documents
  - Task-Based Documentation (How-To Documents)
  - Org Functions
  - Quick Guides
  - Work Categories
- Health Checks and Support Case Library [page 828]

- Health Checks
- Case Documents (Troubleshooting

- Release Information [page 837]
  - Business Configuration Delta Documentation
  - What's New Documentation

10.3.1.2 Documentation Types in Cloud Solutions from SAP

Overview

This document provides a brief overview of the main reference documentation types specific to the SAP cloud solutions.

Business Configuration Documentation

The reference documentation for business configuration covers documentation for the Business Adaptation Catalog (BAC).

- Business Adaptation Catalog
  This reference documentation provides information about scope selection. All scope selection topics and activities to configure the system before and after go live are reflected in elements of the Business Adaptation Catalog (BAC). These elements are business areas, business packages, business topics, and business options. BAC elements require reference documentation if they have one of the following visibility options:
  - *Always*
  - *Scoping Visible*
  - *Hidden* with option type *Blueprint Only*
- Activity List
  This reference documentation provides information about configuring the system before and after go live. In the activity list, reference documentation covers fine-tuning activities and basic data activities.
  - Fine-Tuning
    These documentation types provide users with the information they require to perform activities in the *Fine-Tune* phase of the *Activity List*. These documents can also include information relevant to before and after the completion of a fine-tuning activity. Depending on the complexity of the activities and the activity type, there are three types of documents that can be written for these activities:
    - Fine-Tuning Business Background
      This documentation type describes the activity and guides the user through the activity by listing the required steps and settings. This documentation type can be used for simple activities that lead directly to a configuration screen.
    - Configuration Guides
      This documentation type is suited to activities that are fact sheets. Fact sheets are a collection of related links and texts for further navigation, for example to configuration screens.
  - Basic Data

This documentation type provides information about activities during the *Integrate and Extend* phase of the *Activity List* in business configuration. These activities involve migrating or maintaining core master data as a prerequisite for other fine-tuning, migration, or testing activities. The purpose of this documentation type is to advise a user how to perform an activity and why an activity must be carried out during the configuration phase. Detailed instructions for actually carrying out the steps within the activity can come in the form of links to other existing documents.

For information on how this documentation is structured, see Structure of Reference Documentation for Business Configuration [page 826].

Work Center Documentation

The reference documentation for work centers covers documentation for the following:

- Analytics
  This reference documentation provides information about analytic content that is displayed on the user interface. This includes the following types of objects:
  - Data sources: multidimensional analytical views (MDAVs), key figure structures, and key figure grids.
    Overview of data sources documentation provides information about the multidimensional analytical view (MDAV), key figure structures, and key figure grids that make data available for reports in an area, such as Customer Relationship Management. This type of documentation is available in the guided quick activity for creating reports and is linked in documentation related to reports.
  - Reports
    Report documentation provides information about a compilation of data for evaluation that can be presented in the form of a table, list, text, graph, or combination of these. It is used in detailed scoping and fine-tuning, as well as by users in the application. In scoping, the Overview keyblock of the documentation is displayed. In application Reports work center views, the full report documentation is displayed.
    Overview of reports documentation provides information about the reports that are available in an area, such as Customer Relationship Management by report category. It also contains information about relevant variables in the area. The embedded reports available in the super area are also listed.
    Overview of Reports documentation is available in the Reports View of work centers.
- Business Task Management
  This reference documentation provides information that enable them to understand and process a particular business task management (BTM) task type effectively. Each task type document explains the purpose and business process context of a particular task type in the system, why a particular user receives the task, what kind of response is required, what the follow-up process steps are, and which elements can be configured.
- Business Background Essentials
  This reference documentation covers the concepts and processes that form the SAP cloud solution. Therefore, a differentiation is made between the following types:
  - Concept
    Concepts cover a general business concept or technical concept that has a particular implementation in the SAP cloud solution. A concept can be specific to one view or can span several views, business areas, or even application areas.
  - Process
    Processes cover a sequence of logically related events or steps that have a specific outcome or result in the SAP cloud solution. A process can be specific to one particular view or can span several views, business areas, or even application areas.

- Tasks (How-To Documentation)
  This reference documentation provides a user with the step-by-step information they require to be able to perform a task in the SAP cloud solution. Typically, it also includes information that the user needs to know before and after the completion of a task.
- Org Functions
  An org function is a predefined entity in the SAP cloud solution that denotes functional responsibility for aspects of one or more business processes. In Organizational Management (OM), the solution administrator assigns org functions to org units in order to define the role of each org unit within the context of the organization's overall business operation. The org function assignment for a given org unit influences work center proposals for users assigned to this org unit. Each org function document describes a specific org function in terms of the associated business processes, business documents, and work centers. The purpose of org function documentation is to support the OM administrator who is responsible for modeling the org structure and assigning org functions to the org units.
- View Quick Guide
  A quick guide provides a user with a "quick" overview of a view. It is also the central entry point for accessing more detailed information related to the view. That is, for a more in-depth understanding, the user has the option of reading the documents linked from the quick guide. It provides the following key elements:
  - Summary of what the view is used for
  - Business background
  - Tasks that can be performed
- Work Categories
  A work category (also known as responsibility) represents a type of work related to an org function. It contains rules that determine which set of work items is distributed to which org unit or employee. This is known as the process of work distribution, where a work item can be any business document or a task in Business Task Management (BTM).
  The distribution of work items is usually defined by the access rights that are given to a user. However, using work categories, work items can be directly assigned to a specific org unit or employee. Each work category provides a set of parameters that can be used to define work distribution rules for the related work items.
  The SAP cloud solution provides specific work categories for organizational work distribution and employee work distribution.
  Each work category document describes a specific work category in terms of the associated business process, business documents, tasks, parameters, and work distribution rules. The primary purpose of work category content is to support solution administrators who are responsible for effective distribution of work in the system.

For more information on how this documentation is structured, see Structure of Reference Documentation for Work Centers [page 830].

Health Checks and Support Case Library

This reference documentation covers the following:

- Health Checks
  A health check is an incident generated by the system when functions or applications are not operating properly. The reference documentation provides information about the cause of the incident and the information required to resolve the issue.

- Support Case Library (Troubleshooting)
  This reference documentation provides information about the symptoms, cause, and resolution of system issues that are reported as incidents in the support cockpit. Each case document covers one specific issue which may in turn be relevant to one or more related incidents.
  Case documents provide a practical, searchable medium for quickly identifying solutions to system issues and answers to questions raised by customers. Their purpose is to reduce the number of reported incidents by making solutions readily available.
  Case documents address the following types of issues:
  - How-to explanations
  - Assistance with changes to business configuration or master data
  - Workarounds
  - Explanations of system behavior

For more information on how this documentation is structured, see Structure of Reference Documentation for Health Checks and Support Case Library [page 828].

Release Information

This reference documentation includes the following:

- Business Configuration Delta Documentation
  Business Configuration (BC) Delta documentation provides information on all changes that have taken place since the last release in the configuration of the SAP cloud solution that impact customer settings. The changes usually refer to settings that the customer performed during scoping and to settings made from the Activity List. These scope selections and activities are reflected in elements in the business configuration catalog (BAC). The BAC elements are known to the as business configuration elements.
- What's New Documentation
  What's New documentation is a series of documents that, taken as a whole, describes what is new and what has been changed or deleted in a particular version of the SAP standard cloud solution. The first document in the series describes system-wide changes. Depending on the level of change within each area and its parts, other documents may be created for specific areas (for example, Customer Relationship Management), topics (for example, Organizational Management), or views (for example, Opportunities).

For more information on how this documentation is structured, see Structure of Release Information [page 837].

10.3.1.3 Structure of Reference Documentation for Business Configuration

Overview

The reference documentation for business configuration covers Business Adaptation Catalog (BAC) documentation and Activity List documentation:

Business Adaptation Catalog Documentation

This documentation provides information about scope selection. All scope selection topics and activities to configure the system before and after go live are reflected in elements of the Business Adaptation Catalog (BAC). These elements are business areas, business packages, business topics, and business options. BAC elements require reference documentation if they have one of the following visibility options:

- *Always*
- *Scoping Visible*
- *Hidden* with option type *Blueprint Only*

Structure of BAC Reference Documentation (Excluding Business Options for Reports)

Keyblock	Answers the Following Questions
Title	What is the title of the BAC element?
Overview	What function does this element provide? What are the benefits of using this element?
Relevance	When or why should the user select this element? What are the implications of selecting or not selecting this element? Has this element been selected by constraint?

Structure of BAC Reference Documentation — Business Options for Reports

Keyblock	Answers the Following Questions
Title	What is the title of the report?
Overview	What can the user do with the report? Why should the user use the report?

> **i Note**
>
> The documentation for business options for reports matches that of the related report documentation. For more information, see Structure of Reference Documentation for Work Centers [page 830].

Activity List Documentation

This reference documentation provides information about configuring the system before and after go live. In the activity list, reference documentation covers fine-tuning activities and basic data activities.

Fine-Tuning Documentation

These documentation types provide users with the information they require to perform activities in the *Fine-Tune* phase of the *Activity List*. These documents can also include information relevant to before and after the completion of a fine-tuning activity. Depending on the complexity of the activities and the activity type, the following types of documents can be written for these activities:

- Activity Documentation
  This documentation type describes the activity and guides the user through the activity by listing the required steps and settings. This documentation type can be used for simple activities that lead directly to a configuration screen.

Structure of Activity Documentation

Keyblock	Answers the Following Questions
Title	What is the title of the configuration activity?
Overview	What does this configuration activity involve (in summary)? What are the implications for the user's business?
Prerequisites	What other configuration activities have to be performed **before** this activity?
Steps	What are the steps that the user must carry out to perform the activity?
Follow-On Activities	What other configuration activities have to be performed **after** this activity?

- Configuration Guide Documents
  This type of document provides a starting point for completing activities contained in an activity details screen (otherwise known as a fact sheet). It is also the central entry point for accessing more detailed information related to the activity. That is, for a more in-depth understanding, the user has the option of reading the documents linked from the configuration guide.

Structure of Configuration Guide Documentation

Keyblock	Answers the Following Questions
Title	What is the name of the configuration activity, followed by "- Configuration Guide"? For example, "Suppliers — Configuration Guide".
Overview	What are the purpose and result of this configuration activity? Where is this activity located in the system?
Prerequisites	What prerequisites must be fulfilled before completing this activity? Are there any other activities which must be completed **before** this activity can be completed?

Keyblock	Answers the Following Questions
Business Background	What are the key concepts related to this activity?
Tasks	For each individual configuration activity: • What is the title of the configuration activity? What specific action do I need to take to perform this configuration activity? In which sequence should I perform these steps? (If applicable, also explain what the result of each successive action is.)
Follow-Up Activities	What other configuration activities have to be performed **after** this activity, if any, and where do they take place?

Basic Data Documentation

This documentation type provides information about activities during the *Integrate and Extend* phase of the *Activity List* in business configuration. These activities involve migrating or maintaining core master data as a prerequisite for other fine-tuning, migration, or testing activities. The purpose of this documentation type is to advise a user how to perform an activity and why an activity must be carried out during the configuration phase. Detailed instructions for actually carrying out the steps within the activity can come in the form of links to other existing documents.

Structure of Basic Data Documentation

Keyblock	Answers the Following Questions
Title	What is the title of the Basic Data activity?
Overview	What does this activity involve (in summary)? Why must the user perform this activity during configuration?
Prerequisites	What other activities have to be performed **before** this activity?
Steps	What are the steps the user has to perform in this activity?
Follow-Up Activities	What other activities have to be performed **after** this activity?

10.3.1.4 Structure of Reference Documentation for Health Checks and Support Case Library

Overview

This document provides a brief overview of the reference documentation types specific to Health Checks and the Support Case Library (troubleshooting documentation).

Health Check Documentation

A health check is an incident generated by the system when functions or applications are not operating properly. The reference documentation provides information about the cause of the incident and the information required to resolve the issue.

Structure of Health Check Documentation

Keyblock	Answers the Following Questions
Title	What is the name of the error/incident?
Description	What happened in the system/application?
Recommended Action	What steps need to be taken to solve the problem?

Support Case Library (Troubleshooting) Documentation

This reference documentation provides information about the symptoms, cause, and resolution of system issues that are reported as incidents. Each case document covers one specific issue which may in turn be relevant to one or more related incidents.

Case documents provide a practical, searchable medium for quickly identifying solutions to system issues and answers to questions raised by customers. Their purpose is to reduce the number of reported incidents by making solutions readily available.

Case documents address the following types of issues:

- How-to explanations
- Assistance with changes to business configuration or master data
- Workarounds
- Explanations of system behavior

Structure of Support Case Library Documentation

Keyblock	Answers the Following Questions
Title	What is the name of the issue?
Short Description	What is the issue?
Description	What are the symptoms of the issue? What is the context of the issue? What are the implications?

Keyblock	Answers the Following Questions
Reproducing the Issue	Can you outline the events or actions that led to the issue? How can the issue be reproduced?
Cause	In administrator terms, what is the cause of the issue? What is the technical background?
Resolution	What can a support consultant or administrator do to address the issue?
See Also	Where can the user find more information? (cross-reference)

10.3.1.5 Structure of Reference Documentation for Work Centers

Overview

The reference documentation covers documentation for the following:

- Analytics
- Business Task Management
- Business Background Essentials
- Tasks (How-To Documentation)
- Org Functions
- Work Categories

Analytics Documentation

This reference documentation provides information about analytic content that is displayed on the user interface. This includes the following types of objects:

- Data sources: multidimensional analytical views (MDAVs), key figure structures, and key figure grids. Overview of data sources documentation provides information about the multidimensional analytical view (MDAV), key figure structures, and key figure grids that make data available for reports in a particular area, such as Customer Relationship Management. This type of documentation is available in the guided quick activity for creating reports and is linked in documentation related to reports.

Structure of Data Source Documentation

Keyblock	Answers the Following Questions
Title	For which super area is the overview of data sources? For example, Overview of Data Sources for Customer Relationship Management

Keyblock	Answers the Following Questions
Overview	What is the purpose of the document?
Available Data Sources	What MDAVs are available in the super area?
Selected Data Sources	Do any MDAVs, key figure structures, and key figure grids require detailed documentation?
Available Key Figure Groups	What key figure structures and key figure grids are available in the area?
Common Characteristics and Key Figures	Which basic characteristics and key figures are available that are used in multiple reports?
See Also	Which additional documentation exists? (cross-reference)

- Reports
  Report documentation provides information about a compilation of data for evaluation that can be presented in the form of a table, list, text, graph, or combination of these. It is used in detailed scoping and fine-tuning, as well as by users in the application. In scoping in the business options for reports, the Overview keyblock of the documentation is displayed. In the Reports views, the full report documentation is displayed.

Structure of Report Documentation

Keyblock	Answers the Following Questions
Title	What is the name of the report?
Overview	What can the user do with the report? Why should the user use the report?
Prerequisites	Are there any prerequisites to running the report?
Views	Which views are available with the report?
Features	Are all of the following document subheadings relevant for the report? • **Running the Report**: Which variables are most important? • **Analyzing the Report**: What can the user see in the (initial state of the) report? Is the data filtered? Which characteristics can the user use to filter and analyze the content of this report further? Which elements of the report provide the possibility of opening or navigating to other related reports or business object interfaces, such as Overviews (fact sheets)?
See Also	Which additional documentation exists? (cross-reference)

- Overview of Reports
  Overview of reports documentation provides information about the reports that are available in a particular area, such as Customer Relationship Management by report category. It also contains information about relevant variables in the area. The embedded reports available in the super area are also listed. Overview of Reports documentation is available in the Reports view.

Structure of Overview of Reports Documentation

Keyblock	Answers the Following Questions
Title	For which area is the overview of reports? For example, Overview of Reports for Customer Relationship Management
Overview	What is the purpose of the overview?
Reports	Which reports are associated with which report category?
Common Variables	What are the basic variables relevant for reports?
Embedded Reports	Are there any information consumer patterns (ICPs)?
See Also	Which additional documentation exists? (cross-reference)

Business Task Management Documentation

This reference documentation provides information that enables users to understand and process a particular task type effectively. Each task type document explains the purpose and business process context of a particular task type in the system, why a particular user receives the task, what kind of response is required, what the follow-up process steps are, and which elements can be configured.

Structure of BTM Task Type Documentation

Keyblock	Answers the Following Questions
Title	What is the name of this task type?
Overview	What kind of task type is this and why was it created?
Relevance	Why has this task type appeared in the user's worklist(s)? To which areas of business is it relevant?
Response	What are the possible actions I can take in responding to this task? What is the result of each possible action, and what are the business and workflow implications?
Configuration	Is it possible to configure this task type?
See Also	Where can the user find more information? (cross-reference)

Business Background Documentation

This reference documentation covers the concepts and processes that form the SAP cloud solution. Therefore, a differentiation is made between the following types:

- Concept Documents
  Concept documents cover a general business concept or technical concept that has a particular implementation in the SAP cloud solution. A concept can be specific to one or more views or can span several business areas or even application areas.

Structure of Concept Documents

Keyblock	Answers the Following Questions
Title	What is the name of the concept?
Overview	When and why is the concept relevant for the target group? What is the business context? Is the concept country or industry-specific? What does the target group need to know about the concept?
Prerequisites	Are there any steps or settings (for example, business configuration activities) that need to be carried out before execution of this concept is possible?
Freestyle Subheadings	Which keyblock header(s) are needed to structure the document in a meaningful way?
See Also	Where can the user find more information? (cross-reference)

- Process Documents
  Process documents cover a sequence of logically related events or steps that have a specific outcome or result in the SAP cloud solution. A process can be specific to one or more views, or can span several business areas or even application areas.

Structure of Process Documents

Keyblock	Answers the Following Questions
Title	What is the name of the process?
Overview	When and why is the process relevant for the target group? What is the business context? Is the process country or industry-specific? What does the target group need to know about the process? Is the process mainly user-driven or system-driven?
Prerequisites	Are there any steps or settings (for example, business configuration activities) that need to be carried out before this process can be executed?

Keyblock	Answers the Following Questions
Process Flow	What are the steps involved in the process? Are there any variants of the process?
See Also	Where can the user find more information? (cross-reference)

How-To Documentation

This reference documentation provides a user with the step-by-step information they require to be able to perform a task in the system. Typically, it also includes information that the user needs to know before and after the completion of a task.

Structure of How-To Documentation

Keyblock	Answers the Following Questions
Title	What is the name of the task? Precede the activity name by a verb, for example Create Opportunity.
Overview	What are the purpose and result of this task? Why or when should the user perform this task? Does the user need any additional background information to help them understand the concept behind the task?
Prerequisites	What does the user need to do before performing this task? Which basic settings need to be in place before the user can perform this task? What are the preceding processes or tasks, and where do they take place?
Steps	What specific action does the user need to take to perform this task? In which sequence should the user perform these steps? (If applicable, also explain what the result of each successive action is.)
Result	Does the completion of this task trigger further processes or tasks, or does it have an impact on other processes or tasks? If so, what are these follow-on processes or tasks?
Alternatives	Is there an alternative way of performing this entire task?
Example	Is there an example of this task?
See Also	Where can the user find more information related to this task? (cross-reference)

Org Functions

An org function is a predefined entity in the system that denotes functional responsibility for aspects of one or more business processes. In Organizational Management (OM), the administrator in the SAP cloud solution assigns org functions to org units in order to define the role of each org unit within the context of the organization's overall business operation. The org function assignment for a given org unit influences work center/tab proposals for users assigned to this org unit. Each org function document describes a specific org function in terms of the associated business processes, business documents, and work centers. The purpose of org function documentation is to support the OM administrator who is responsible for modeling the org structure and assigning org functions to the org units.

Structure of Org Function Documentation

Keyblock	Answers the Following Questions
Title	What is the name of the org function?
Overview	To which functional area is the org function related? What are the associated business processes, business documents, and tasks? What are the associated work centers/tabs?
Relevance	From an organizational and operational point of view, when or why should the customer use this org function? What are the implications of not using it? What are the benefits? Are there any unusual use cases?
Configuration	What are the business configuration prerequisites for using this org function?
Constraints	What constraints apply to the use of this org function within the context of the org structure?
Work Distribution	Which work category or work categories are associated with this org function?
Example	Is there an example that shows how this org function can be used?
See Also	Where can the user find more information related to this org function? (cross-reference)

Quick Guide Documentation

A Quick Guide provides a user with a "quick" overview of a view. It is also the central entry point for accessing more detailed information related to the view. That is, for a more in-depth understanding, the user has the option of reading the documents linked from the Quick Guide. The Quick Guide provides the following key elements:

- Summary of what the view is used for
- Business background
- Tasks that can be performed

Structure of Quick Guide Documentation

Keyblock	Answers the Following Questions
Title	What is the name of the view followed by “Quick Guide”? For example, “Opportunities Quick Guide”.
NO HEADER	What is the view used for in summary?
Business Background	What are the key concepts and processes? What are the related business scenario(s)?
Tasks	To which business area does this belong? What are the tasks that the user can perform?

Work Category Documentation

A work category (also known as responsibility) represents a type of work related to an org function. It contains rules that determine which set of work items is distributed to which org unit or employee. This is known as the process of work distribution where a work item can be any business document or a task in Business Task Management (BTM).

The distribution of work items is usually defined by the access rights that are given to a user. However, by using work categories, work items can be directly assigned to a specific org unit or employee. Each work category provides a set of parameters that can be used to define work distribution rules for the related work items.

The system provides specific work categories for organizational work distribution and employee work distribution.

Each work category document describes a specific work category in terms of the associated business process, business documents, tasks, parameters, and work distribution rules. The primary purpose of work category content is to support administrators who are responsible for effective distribution of work in the system.

Structure of Work Category Documentation

Keyblock	Answers the Following Questions
Title	What is the name of the work category?
Overview	Is the work category relevant to organizational work distribution or employee work distribution? What are the associated business processes and business documents? What are the related work centers/view/tabs?
Prerequisites	What are the prerequisites for using this work category? Are there any configuration (fine-tuning) activities that are relevant for this work category? Which org function must be assigned?

Keyblock	Answers the Following Questions
Work Category Parameters	Which parameters are used to determine the responsibility?
Work Distribution	How does this work category support the distribution of work in the system?
Example	Is there an example that shows how this work category can be used?
See Also	Where can the user find more information related to this work category? (cross-reference)

10.3.1.6 Structure of Release Information

Overview

This document provides a brief overview of documenting release information covering changes to the system between releases.

Business Configuration Delta Documentation

Business Configuration (BC) Delta documentation provides information on all changes that have taken place since the last release in the configuration of the system that impact customer settings.

The changes usually refer to settings that the customer performed during scoping and to settings made from the activity list. These scope selections and activities are reflected in elements in the business adaptation catalog (BAC). The BAC elements are known as business configuration elements.

Structure of BC Delta Documentation

Keyblock	Answers the Following Questions
Title	What is the BC Delta called? Use the following guidelines for the title: • For a new BAC element, use “New: <Name of BAC Element>” • For a changed BAC element, use “Changed: <Name of BAC Element>” • For a deleted BAC element, use “Deleted: <Name of BAC Element>”
Relevance	What elements did the customer select in order to include this element in their scope?
Overview of Changes	What is the current setting? What is the issue or challenge that needs to be addressed? What is the solution?

Keyblock	Answers the Following Questions
Details	What are the new or changed features? What is the affect of this change on the behavior of the system? Is there additional information that may be relevant to the customer?
Required Actions	What are the activities that are required from the customer?

What's New Documentation

What's New documentation is a series of documents that, taken as a whole, describes what is new and what has been changed or deleted in a particular version of the standard system. The first document in the series describes system-wide changes. Depending on the level of change within each area and its parts, other documents may be created for specific areas (for example, for Customer Relationship Management), topics (for example, Organizational Management), or views (for example, Opportunities).

Structure of What's New Documentation

Keyblock	Answers the Following Questions
Title	To which area of the solution does this information apply?
Overview	To which version and area of the solution does this information apply? Which topics are covered?
What Is New	What is the new development or function? What are the business benefits of using the new development or function for the customer? How does the new development or function affect the user interface (for example, fields, menu paths, and screen title)? Only mention implications that are relevant for the customer. What actions must the customer take to use the new function (for example, in business configuration)? How does the new development or function affect existing data such as master data or transactional data? What is the impact of this new development or function on other areas of the solution (for example, on business processes or scenarios)?

Keyblock	Answers the Following Questions
What Has Changed	What is the changed development or function? What are the business benefits of using the changed development or function for the customer? How does the changed development or function affect the user interface (for example, fields, menu paths, and screen title)? Only mention changes that are relevant for the customer. What actions must the customer take to use the changed function (for example, in business configuration)? How does the changed development or changed function affect existing data such as master data or transactional data? What is the impact of this change on other areas of the solution (for example, on business processes or scenarios)? Which relevant terms have been changed?
Country-Specific Changes	What is the new/changed development or function for a specific country? What are the business benefits of using the new/changed development or function for the customer? How does the new/changed development or function affect the user interface (for example, fields, menu paths, and screen title)? Only mention changes that are relevant for the customer. What actions must the customer take to use the new or changed function (for example, in business configuration)? How does the new development or changed function affect existing data such as master data, or transactional data? What is the impact of this change on other areas of the solution (for example, on business processes or scenarios)? Which relevant terms have been changed?
See Also	Where can I find further information about related topics (for example, related BC Delta documents)?

11 Copyright / Legal Notice

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. Please see http://www.sap.com/corporate-en/legal/copyright/index.epx for additional trademark information and notices.

icon courtesy of W3C, see http://creativecommons.org/licenses/by/3.0/ .

12 Glossary

This glossary defines terms that you will encounter while working in the studio and with the product documentation.

access control	A method of allowing and restricting user access rights for a work center view based on the access control list of a business object. For example, instance-based access control can be used to allow access to company-specific data only.
access control list	A list of access groups that is needed to perform authorization checks for business object instances.
action	A script file that executes business logic at a particular point in time within the business object's lifecycle. Actions can be triggered from the user interface or from events or other actions.
action code	A coded representation of an instruction to the recipient of an XML message. For example, 01 is the action code for Create, 02 for Change, and so on.
activity list	A structured collection of mandatory and critical tasks, generated from the customer's scoping, that must be completed either before the solution can go live for the first time or before changes can be implemented later in the solution life cycle. The activity list may also include optional activities that can be completed at any stage.
add-on solution	A solution that enables the implementation of additional business logic in an SAP solution and the development of enhanced functions for business users.
Administration toolbar	A set of functions that allows administrators to manage solutions.
administrator	A system user who has administrator rights in an SAP cloud solution. You use this user in the studio if you want to define analytics content or create a mashup. To use the administrator user, you enable admin mode for a solution in the studio.
advanced search	A search that allows the user to combine free text search and attribute search. The user can specify the search, display, and filter options to locate the item they require.
alert	A task with the highest available priority that requires immediate user action.
alternative key	An annotation with which you define a unique key of a business object node. You can it in an association and it is stored as an additional key field in data searches and queries.
Analytics	Analytics provides you with the necessary infrastructure for the analysis of your enterprise. A set of predefined reports from all business areas is available that cover the most important reporting needs.

anchor	A stable point to which any user interface modifications to SAP floorplans are attached.
Application and User Management	A work center that allows administrators to perform application administration tasks and to manage access rights.
application log	A support mechanism for mass data runs. Mass data runs run in the background. No direct feedback is provided to the user on what exactly has happened is given, so the application log allows users to retrace the results.
application-to-application communication	A process by which messages containing information about updates to business documents are sent between process components within the system. For example, when creating a sales order, the system updates related business documents within the system.
approval notification	An electronic message sent to the requester to indicate that approval has been granted or rejected by the authorized approver.
approval process	One or multiple steps by which a business document is addressed to one or more employees responsible for approval, including any defined conditions.
Approval Processes	A work center view in which you can display, edit, copy, and delete versions of approval processes related to business documents.
approval status	An indicator of the step reached in the approval workflow of an object, for example, an invoice that has to be reviewed by an authorized approver could be listed as "waiting for approval".
approval task	A part of an approval process. Approval tasks are approval, approval/rejection notifications, and revision.
Approval Wizard	A guided activity for creating a default approval process for a custom business object in the studio.
assigned work centers	A work center which a user can access to do their work.
association	A relationship between two business object nodes.
asynchronous communication	A form of electronic communication that does not take place in real time. E-mail is an example of electronic asynchronous communication.
availability	The capacity of a service or a component to fulfill its required function at a specific moment or over a set time period. Usually, this is measured using an availability ratio, which compares the length of time that the service has been available with the duration agreed with the customer.
BAC element	A project item for business configuration such as a business option or a business topic that is required to activate a solution in a customer system. In the standard SAP solution, a BAC element is a scoping element and can refer to any level of the business adaptation catalog hierarchy, including business packages, business topics, and business options.

BO Browser	Tab of the BO Browser/Data Model tool window in the UI designer. It displays a business object as a tree structure with all available nodes and fields.
browser-based report	A type of report that can be viewed in a Web browser.
build	The version of a studio installation.
business adaptation catalog	A central structure that uses business language to describe and organize all the capabilities of SAP solutions. It structures the capabilities into a hierarchy of business areas, packages, topics, and options. Customers and prospects make selections from the catalog based on their specific business needs. The catalog enables change both during the initial setup of the solution and throughout its life cycle.
Business Analytics	A work center that enables key users to create and assign reports, and to create and edit key performance indicator evaluations.
business area	The first and highest level in the hierarchy of the business adaptation catalog. It represents the collective group of functions that the solution supports in a specific part of the company. Examples are sales and human resources. Each business area contains a number of business packages.
business configuration	The process of adapting the solution to the day-to-day business requirements of the customer at any time in the life cycle of the solution without the need for technical skills.
Business Configuration	A work center where users can configure the solution to meet their specific business requirements. You can access the Business Configuration work center both before and after initial go live, but different functionality is available to you depending on what stage you are at in configuring your solution.
business configuration content	Project items that are anchored in the business adaptation catalog to allow key users to implement a solution in the production system. Examples include business options and business topics, business configuration sets, and business configuration views.
business configuration object	A project item for business configuration that represents a configuration table and is used to create business configuration sets. A business configuration object is a business object category.
Business Configuration Object Wizard	A guided activity for creating a business configuration object that is then used to create a business configuration set.
business configuration set	A technical container for the configuration settings and corresponding values that are associated with a business option. The content in a BC set is deployed to the runtime environment where it is loaded into the underlying configuration tables. Each configuration setting in a BC set has attributes that specify whether it is hidden, visible, or changeable for customers.

Business Configuration Set Wizard	A guided activity for creating a business configuration set based on a business configuration object.
Business Configuration Wizard	A guided activity for creating business options and business topics and anchoring this business configuration content in the business adaptation catalog.
business document	A representation of a business object that has a definite format and contains structured information necessary for business transactions. Examples include invoices, quotes, proposals, requests, purchase orders, and production orders.
business object	A model of a real-world object, for example, an employee or a sales order in business application systems.
business object extension	An SAP business object that has been extended to include additional fields.
business object instance	A representation of the object type of a business object. For example, the sales order numbers 12345 and 56789 are instances of the Sales Order business object type.
business option	The fourth level in the hierarchy of the business adaptation catalog. Business options can be hidden or can appear in scoping, fine-tuning, and/or the solution proposal. It represents the most detailed decision level involved in defining or adapting a solution capability. Each business option specifies a particular way to perform a function. It is based on predefined content. Examples are General Ledger Accounting and Customers. Each business option belongs to a business topic.
business package	The second level in the hierarchy of the business adaptation catalog. It covers the organization, processes, management support and master data of a given application area. Each business package belongs to a business area and contains a number of business topics.
Business Task Management	An integral part of the system which manages the prioritization and assignment of tasks to the appropriate users.
Business Task Management	A work center view that enables users to monitor unassigned items and automated tasks with errors.
business topic	The third level in the hierarchy of the business adaptation catalog. It relates to specific functions within an application area. Examples are Account Management and Time Administration. Each business topic belongs to a business package and contains a number of business options.
business user	A user that can log on to the system. You grant access rights to users by assigning work centers or work center views to their business user.
business-to-business	Electronic transactions between one business and another, for example, between a supplier and a retailer. Business-to-business transactions are typically high volume and involve a long-standing business relationship.
calculated key figure	A key figure that is determined using calculation rules or formulas.

change transaction	An action that allows you to modify an SAP floorplan. For example, you can use a change transaction to add a view to a standard work center.
characteristic	A field according to which values are selected. Characteristics are alphanumeric, numeric, or text values. Examples include Product ID, Supplier, and Purchase Order Status.
chart of accounts	A systematic listing of all general ledger accounts used by an organization of one or more companies, normally in alphanumeric order.
clarification request	A manually created task used for asking another user for more information on an item.
cloud solution	Software hosted by SAP, for example, SAP Business ByDesign.
code list	A field with a set of predefined values that is of data type Code.
Code List Data Type Wizard	A guided activity for creating a static code list in the studio.
collection	A table that is defined on the basis of business object nodes or node elements, for example, a collection of sales order items.
combined data source	A type of data source that combines two or more data sources. The data sources of which the combined data source consists usually have a set of common characteristics but semantically different key figures.
common task	A task that you perform frequently, such as creating a sales order.
communication arrangement	An agreement between an organization and a customer, supplier, or partner to exchange business documents electronically.
communication scenario	A set of predefined services for a business process that can be used to create a communication arrangement.
Communication Scenario Definition Wizard	A guided activity for defining a communication scenario in the studio.
communication system	A system that is used for application integration between the solution and an internal or external system. Communication systems can be, for example, external time recording or master data systems.
compliance	The process of conforming to a specification, standard or law, for example a law relating to data protection.
composition	A navigational relationship from a superordinate to a subordinate node and vice versa: Composition associations point from the parent to a child node and from the child to the parent node. They are automatically part of the business object model when a node is created.
condition	A predefined filter to restrict the data displayed to a specified characteristic and key figure. There are two types of conditions: simple and ranked.

configurable code list	A code list whose values can be changed by the customer in the scoping phase of SAP's cloud solution. A configurable code list is created by defining a BC set for a business configuration object and selecting the "Create Code List Data Type" option.
Configuration Explorer	A tool window in the UI designer that displays the SAP repository content.
content area	The area of the application window for performing user tasks. The content area is visible on every page and changes when the user navigates from one page to another.
continuous improvement	A commitment by SAP to regularly analyze all aspects of a customer system configuration and then identify improvement potential and recommend actions. The analysis covers a range of issues including KPIs and bottlenecks. Actions SAP can recommend include changing system settings and installing delivery patches.
control	A component of a floorplan such as an input field or a button.
Controller	A tab in the content area of the user interface designer. On the Controller tab, you define the interaction between user interface components by creating event handlers, interfaces, navigations, and queries.
Core Component Technical Specification	Describes a methodology for semantic data modeling that achieves a common understanding of data message types on a syntax independent level. Developed by UN/CEFACT and ISO Technical Committee.
Create Form Wizard	A guided activity for creating print forms, defining form properties, and defining country and language-specific form variants.
custom business configuration object	A business configuration object created as part of a solution in the software development kit.
custom business object	A business object defined as part of a solution in the software development kit.
custom control	A user interface control created in Microsoft Visual Studio using Microsoft Silverlight. The control can be used in the studio to enhance screens in the SAP solution.
custom pane	A user interface pane created in Microsoft Visual Studio using Microsoft Silverlight. The control can be used in the studio to enhance screens in SAP solutions.
custom query	A query created as part of a solution in the software development kit.
custom reuse library	A collection of user-defined functions. You can use a library to organize the code in your solution or to reuse code in several script files.
customer-specific field	An extension field in a cloud solution.
customer-specific solution	A partner solution that the partner designs and develops for a specific customer.
Data Model	Part of the BO Browser/Data Model tool window and tab in the content area of the user interface designer. The Data Model displays the data structure of the business

	object. You can edit the structure and bind the data model elements to the business object elements.
data source	An object containing key figures and characteristics, which provides a multidimensional view of business data for reporting.
Data Source Wizard	A guided activity for creating and defining a data source. In addition, you can define transformation rules, define aggregation behavior for key figures, and reference other data sources.
default approval process	A process created in the studio for a custom business object for use in an SAP solution. A default approval process has one approval step, no conditions, and unlimited validity.
dependent object	A reusable part of a business object that cannot stand alone but can only be used in the context of the business object.
deploy a solution	Upload a solution to a customer system or to a test system.
deploy business configuration	Make business configuration content created in the studio available in the system.
deployment unit	A piece of software that can be operated on a separate system isolated from other pieces of software. All deployment units can communicate directly with the Foundation deployment unit.
derived name	The default naming convention provided by the software development kit to name a message data type in a Web service operation. A derived name is based on the names of the service operation and the business object element on which the service operation is allowed to be performed.
Design Data Sources	A work center view in which you can create, edit, and delete data sources and create reports.
Designer	A tab in the content area of the user interface designer. On the Designer tab, you design the user interface of an application with the controls available in the Toolbox.
Design Reports	A work center view in which you can create, edit, delete, and assign reports to work center views.
details area	An area of the screen providing details about a user selection. For example, if a user selects a row in a work list, the details area provides details about that row.
developer desktop	The user interface of the studio that presents and provides access to development tools in an integrated development environment (IDE).
dialog box	A box on the screen that a user must respond to before resuming work. Sometimes the user prompts the dialog box to display and sometimes it displays without prompting.
dimension	A criterion by which characteristics are categorized.

dropdown list	A list of entries that a user can display by clicking a down-facing arrow to the right of the first entry in the list.
element	A component of the business adaptation catalog. An element is a general term that can refer to any level of the catalog hierarchy, for example, a business package, topic, or option.
element	Part of a business object. Elements are displayed as fields on the UI.
embedded component	A reuseable user interface component that can be added to other floorplans.
embedded report	A report that is embedded in a specific context, such as in an overview document.
Employee Work Distribution	A work center view that enables the key user to define users responsible for specific work items.
endpoint	A location for accessing a Web service using a specific protocol and data format.
enhanced controller object	A special business object created only for user interface scenarios. It is needed to implement certain functions and message handling. It is also used when two business objects have to interact in a way that is not declarable using existing tools.
enhancement implementation	An extension of an enhancement option that allows customers to apply business rules that meet specific requirements such as country-specific requirements.
enhancement option	An entity provided by SAP to allow the standard behavior of a specific application, such as Financials, to be enhanced without modifying the standard solution.
enhancement option type	An attribute that indicates whether an enhancement option is for single use or multiple use.
enterprise search	An application that allows you to locate and view data stored in your system. You can search for structured, for example purchase orders, and unstructured data, for example Microsoft Word documents. This data is organized into the following categories: Files, People, Reports, and Business Objects.
error message	A message informing a user about an error. The message typically explains what went wrong, why it went wrong, and what the user can do about it. Most error messages display in the message area of the screen. However, if the system requires that the user must respond to an error, the error message displays in a dialog box.
event	A script file that executes business logic at a particular point in time within the business object's lifecycle, for example, before save or after modify.
exception	A deviation from a defined threshold value or interval. Exceptions can be applied to data, results, or a combination thereof and can be displayed as an arrow, background color, or traffic light.
exception aggregation	Aggregation behavior, which deviates from standard aggregation (such as summation), for values of a calculated key figure. The deviation is only valid for a specified characteristic.

extendable node	A node of an SAP business object that can be enhanced with additional fields, the so-called extension fields.
Extensibility Explorer	A tool window that displays the anchors attached to a floorplan.
extension field	A field that has been added to an SAP business object. Extension fields can be used, for example, in screens, forms, reports, and search categories.
extension scenario	A representation of all business contexts that are part of a process flow in which an extension field may be involved.
External Web Service Integration Wizard	A guided activity for integrating an external Web service into the partner solution using SOAP or REST.
fact sheet	A fact sheet is a one-page read-only overview of an object. It displays the object properties, can have different views, and can contain links for navigating to related objects. You can model this overview in the UI designer.
favorites	A location where hyperlinks to commonly viewed content (for example, reports and business documents) or Web pages can be stored and easily accessed.
file input	A general method for uploading external data files on specific interfaces into an SAP solution.
file input run	A mass data run that uploads data files from external applications into the SAP cloud solution, using existing service interfaces.
filter	To retrieve data that matches user-defined filter criteria.
filter	A project item that belongs to an enhancement implementation that allows users to work with a specific set of data.
fine-tuning	The configuration of the solution to meet customer requirements that exceed the level of predefined content in the software. It is based on input and selections made during scoping and occurs before the solution goes live.
fine-tuning project template	A project item for business configuration that contains predefined fine-tuning activities for a specific scenario that is defined by SAP, such as Financials: Chart of Account.
flag	The marking of a business document for later reference, using a flag icon.
foreign key relationship	A relationship between two business object nodes for which the alternative key of the target node is specified.
form data type	A form-specific data type based on an existing data type.
form data type extension	An SAP form data type that has been extended to include additional fields.
Form Template Selection	A work center view that enables key users to create form template rules for business documents.

Form Wizard	A guided activity for creating print forms, defining form properties, and defining country and language-specific form variants.
front-end script	A programming language used in the User Interface Designer to define rules and logic that determine the behavior of user interface (UI) controls. For example, you can write a front-end script that enables an Edit button only when a user selects a field on the UI.
full-text search	Search on an index that contains all words that occur in the indexed documents instead of only abstracts or other metadata.
Gallery	A work center view that allows you to browse reports.
Gantt chart	A graphical display of schedule-related information. Project elements are listed on the left of the chart, the timescale is across the top, and activity durations are shown as date-based horizontal bars.
guided activity	A secondary window consisting of a series of interactive screens that guides you through each step of a particular activity. For example, the New Invoice guided activity guides the user through the steps involved in creating a new invoice.
Guided Activity Floorplan	A floorplan for an activity that can be divided into a logical sequence of steps. It consists of a series of screens that guide the user through an activity to complete a specific task.
HTML mashup	A mashup that integrates HyperText Markup Language (HTML) or JavaScript content as embedded content. Examples include an online map, a widget, or social networking information.
icon	A small pictorial or graphical image that represents an object, action, or status.
inbound service	A Web service that a partner creates based on an SAP business object or a custom business object.
incident trace	A trace that is attached to a support request.
Incidents	A work center view in which a key user resolves incidents or forwards them to the provider for resolution.
information icon	A symbol on the user interface that can be found next to entry fields. When you click on the icon, the system displays useful text.
inport	A port configured to receive a navigation key from the source user interface. This key is used by the target user interface to fill business data.
input field	Element of the user interface. An input field consists of a field name and an input area. You can enter and edit texts in the input area.
installation key	A series of numbers generated by the partner for a multi-customer solution. The installation key enables the partner to upload the solution to a customer tenant.

integrated solution	A solution that enables the integration of applications with SAP's cloud solution. An example is an external Web shop application that creates sales orders in SAP's cloud solution.
IP address	A unique address used to identify a node in an IP network.
item	A business object, record, or piece of information in the system. Examples are Account, Contact, Lead, and feed update.
joined data source	A type of data source that joins two or more data sources. The joined data source consists of a subset of characteristics that are the same in the data sources to be joined along with semantically different key figures.
key figure	A field according to which values are selected. Key figures are numeric values that have a unit of measure or currency assigned. Examples include Invoice Net Value and Purchase Order Quantity.
key figure grid	A group consisting of a key figure structure and a characteristic structure for reporting on two axes.
key figure group	A key figure structure or key figure grid.
key figure structure	A group of predefined key figures for reporting on one axis.
local help	Context-sensitive studio documentation that is displayed in the MS Help Viewer where you can use a full-text search.
local trace	A trace that is run in the system in which it was created.
localization	The process of adapting a product to meet the needs and expectations of a specific country or region, with respect to language, laws, business practices, or culture.
log out	To end a previously authenticated session of the application. When you log out, you can no longer use the application. To use the application again, you must log in again.
maintenance mode	The status of a solution after it has been assembled. You can make changes to the solution in a patch but only restricted changes to certain content types. With these change and delete restrictions, SAP wants to avoid loss of data as well as inconsistencies and errors in the productive customer's solution.
mashup	A Web service or application that has been integrated into an SAP cloud solution. Mashups combine and transform internal business data with information or services provided by external online service providers.
Mashup Authoring	A work center view in which key users or administrators can create and edit mashups.
mashup in-port	The incoming data connection between a screen and a mashup. The mashup in-port is used to input business data from the system into the mashup. This data comes from the screen out-port and can contain multiple parameters.

mashup out-port	The outgoing data connection from a mashup to a screen. The mashup out-port is used to pass the result of the mashup to the system. This is then mapped to the screen in-port.
mashup solution	A solution that enables the integration of Web-based applications or services into the user interface of SAP's cloud solution. An example is the integration of a Web application for maps and route planning into a user interface displaying the address of a business contact.
mashup type	The categorization of mashups by the technology on which they are based. Examples include a URL mashup or HTML mashup.
mashup Web service	A third-party Web service that a key user or an administrator can integrate into an SAP cloud solution so that it can be used by mashups.
Mashup Web Services	A work center view in which key users or administrators can integrate third-party Web services into an SAP solution so that they can be used by mashups.
mass data run	The automatic mass processing of a business transaction. An invoice run is an example of a mass data run.
mass data run object	A business object specifying an automatic mass processing of a business transaction. It contains the selection criteria and the processing results.
Mass Data Run Wizard	A guided activity for creating a mass data run for a custom business object in the studio.
mass-enabled script file	An action or event that can be executed for multiple instances of a business object node.
message definition	A system notification and its attributes defined to assist customers who work with an enhancement implementation.
method tooltip	An on-screen explanation of a method and its parameter definition.
modal dialog	A small window that helps the user to perform a task. It is triggered from a calling application and has to be completed before the user can continue to work in the calling application.
multi-customer solution	A partner solution that the partner can sell to multiple customers.
multiple-use enhancement option	Identifies an enhancement option that can be implemented more than once in the same solution.
My Solutions	A tool window in the studio in which you can create and access solutions in the repository of SAP's cloud solution.
navigation	The means by which a user can move around, within and between, screens. Navigation elements include menus, links, and standard buttons such as Back, Forward, and Exit.

node	A group of elements in a hierarchical structure. For example, a business object can have several nodes that group the business object elements.
non-extendable node	A node of an SAP business object that cannot be enhanced with additional fields, the so-called extension fields.
notification	Defined as part of the default approval process. Notifications are sent to the participants of an approval process to inform them of a task to be carried out. In the studio, you can define an approval notification and a rejection notification.
Notification Rules Wizard	A guided activity for enabling notification rules for a custom business object in the studio.
object-based navigation	A type of user interface interaction. When you create an object-based screen scenario, you create a work center with an object work list, a quick activity floorplan and an object instance floorplan.
object instance floorplan	A combination of UI patterns arranged in a specific order that allows users to create, delete, view, and edit attributes and associations of a business object.
object work list	A list pane that is embedded in a work center view.
online help	Built-in documentation for the studio that is displayed in the Web browser.
operand	A numeric value upon which an operation is performed.
operator	A symbol that describes an operation to take place between two or more values.
outbound service	An external Web service that a partner integrates into the cloud solution.
outport	A port configured to pass the navigation key from the source user interface to the target user interface.
Output Channel Selection	A work center view that enables key users to create output channel rules for the process-integrated output of business documents.
pane	A section of a floorplan.
parameter binding	Mapping of a URL parameter to a screen out-port.
partner solution	A business solution that is designed and developed by an SAP partner to enhance an SAP cloud solution.
password	A combination of alphanumeric characters used by a user to gain access to the system.
patch	Changes or corrections to a customer-specific solution after it has been assembled.
permissions	A set of policies describing whether an identity is allowed to access a certain program or piece of information. The permissions, also called access rights, are grouped into work centers. The work centers can then be assigned to identities, for example, to a user.

personalize	To change or customize something to suit your needs. For example, change the way that your desktop is configured so that you have easy access to the applications you work with.
placeholder	A symbol or chain of symbols that, at runtime, is replaced by text or numbers.
port binding	A grouping of predefined screen out-ports and in-ports that can be used to create a mashup. The port binding defines the type of information that can be used in the mashup and defines on which screens you can use the mashup.
port type package	A blueprint for an inport or an outport. A port type package hosts one or more port types.
process communication error	An error that occurs if the system cannot process a message sent from one business object to another during an update of business data.
Process Communication Errors	A work center view that allows key users to monitor application-to-application and business-to-business communication errors.
process extension scenario	A type of extension scenario that links the data from one business context to other related business contexts. You create an extension scenario from a list of predefined extension scenarios.
production environment	A reserved space that is used by an individual prospect or customer to store all information needed to operate its live solution and run its business. Since any changes that are made here have a direct affect on the running system, the customer typically performs changes and any related testing activities in a separate environment before doing so in the live production environment.
production system	A system used by the customer to work with the solution after it has gone live. This system contains all of the elements required to run the customer's business processes using the SAP cloud solution and can be adapted in the Business Configuration work center using immediate changes and change projects.
project	A container that is used to organize all items of a solution, for example, business objects, screens, and business configuration content.
project item	A file in a project, for example, a business object, a screen, or a form.
Property Explorer	The Properties tool window of the user interface designer. You can view and change properties for each element in the user interface model.
provider	The party responsible for providing IT services or a product to a customer. This can be performed by an in-house provider or a third party.
public solution model (PSM)	Contains all entities in SAP's cloud solutions that are released for external consumers. External consumers can be partners who develop solutions on top of SAP's cloud solutions or administrators who create new reports.
Query Wizard	A guided activity for creating a query, selecting query fields, and defining query parameters.

quick activity	A secondary window consisting of a single interactive screen that allows you to complete a particular activity. For example, the Create Incident quick activity allows you to request support from a key user.
quick activity floorplan	A screen layout that allows users to quickly perform a self-contained or short subtask of a larger task.
quick create floorplan	A preview of detailed information on a thing that a user can see without navigating away from the UI currently displayed. Data shown in a quick create view cannot be edited.
quick view floorplan	A slide-in window, which allows users to edit data without navigating away from the UI currently displayed.
radio button	Element of the graphical user interface. A radio button consists of a round element and a descriptive text. You can use a radio button to select one item only from a list of items.
radio button group	A control in the Toolbox that consists of a UI text and two radio buttons.
refresh	To update a screen, or part of a screen, so that it displays the most current information.
report	A compilation of data for evaluation where online analysis and drill-down is supported in every possible direction and on a very detailed level. Analytical reports are ideal for detailed root cause analysis.
report category	A category by which reports are grouped from a business perspective.
report incident	A function that allows the user to report an incident from any screen within the application.
report wizard	A guided activity with which you can create or change your own reports.
Repository Explorer	A tool window that displays SAP content, such as business objects and data types, which is released with the public solution model (PSM). It also displays the documentation for the different entities.
requester	A person who creates an approval process in SAP's cloud solution.
reset	A function that enables you to return to the original state of a document or application.
resolution	The action taken to solve an incident. Examples of resolutions include patches and work-arounds.
REST Web service	An HTTP-based Web service that is based on REST (Representational state transfer) architectural concepts.
retention time	Period of time data are stored in a system.

reuse library function	A self-contained module of code that operates on values passed to it and typically returns a result.
revision	Task that is part of the approval process. It is sent to the person who created the approval process and informs him or her that the approver has requested changes to the request.
revision task	The task created by the system requiring a user to change data entered because it contains errors or because a change is necessary.
rollover explanation	A brief text that provides instructions or other explanatory information about UI elements, such as fields, buttons, column headers, tab titles, radio buttons, and check boxes, in cases where the label is not self-explanatory. A rollover explanation is displayed when the user points the mouse cursor on the UI.
RSS	A method for publishing content such as news headlines, blogs, or podcasts on the Web using an XML (Extensible Markup Language) file format. The content is automatically updated on a regular basis, and alerts users to new information.
RSS Feed	A document that is either the entire text or the summary of the content of an associated Web site.
RSS/Atom Web service	A Web feed using Really Simple Syndication (RSS) or Atom standards.
SAP Cloud Applications Studio	A software development kit (SDK) that enables SAP partners to adapt and enhance the solution capabilities of SAP's cloud solution. The development tools provided are presented in an integrated development environment (IDE) based on Microsoft Visual Studio.
SAP reuse library	A collection of functions provided by SAP that you can use to perform common tasks such as date conversion or time calculation.
scoping	The process of matching business requirements to the capabilities of the solution, using the business adaptation catalog.
scoping question	An important question the customer answers during scoping to select or deselect a business option from the business adaptation catalog. An example of a scoping question for the Data Migration business topic is “Do you want to migrate data using the built-in migration tool?”
screen out-port	The outgoing data connection from a screen to a mashup. This allows the data from the screen to be passed to a mashup in-port.
script file	A project item that contains business logic for a solution.
scripting language	The programming language used in the studio to define business objects and business object extensions and to implement the business logic.
scripting language compiler	A program that transforms code written in the studio into a language that can be used to execute the code.

search	To look for specific data in business documents, worklists, or the system.
search category	A group of objects with a similar theme that allows users to limit the scope of a search. Categories include Files, Reports, Business Objects, and People, for example employees.
search criteria	The specific values that you look for while doing a search. For example, if you search for all documents created on a certain date, that date is the search criteria.
search result	A list generated after the user specifies the search criteria and executes a search.
section group	An area on the UI that contains information such as fields, links, or lists that belong together semantically.
segregation of duties	A measure used in the prevention of fraud by separating business processes so that more than one person is responsible for completing a task. This is done by assigning access rights and distributing responsibility between several users in the aim of reducing errors and preventing fraud.
service integration	Synchronous or asynchronous message-based communication between a business object and another communication partner, such as a different system, by means of Web-service technology.
service integration definition	A project item containing service integration settings.
Service Integration Wizard	A guided activity for creating message-based communication between a business object and another communication partner, such as a business object in a different deployment unit or a different system.
service level agreement	An agreement with the customer for the assurance of a specific service within a predefined period of time. Service Level Agreements (SLA) define the attributes for service product (for example, maintenance, hotline) that have been agreed upon with the customer in service contracts. The SLA confirms different parameters, such as response time, availability time and system availability. The SLA effects the pricing for each contract item and the date calculation in the service order, to which the relevant contract items refer.
service operation	A specification of a function with a set of message types assigned as signature exposed as a web service. It is grouped by interfaces.
Session Administration	A tool window that displays active user sessions and provides functions for deleting sessions and item locks.
Session Administrator	A tool window that displays active user sessions and provides functions for deleting sessions and item locks.
shortcut	Quick access to a screen without using the common navigation path.
short ID	An identifier used to identify custom work center views in the UI designer and in the SAP cloud solution. You create the short ID in the "Create Screens" dialog in the studio.

single-use enhancement option	Identifies an enhancement option that can only be implemented once in the same solution.
SOAP Web service	A Simple Object Access Protocol(SOAP)-based Web service.
solution	A container that is used to organize all items of a solution, for example, business objects, screens, and business configuration content.
Solution Explorer	A tool window in which you can add, edit, and delete project items.
solution proposal	A document attached to an incident that describes a possible solution to the issue. The processor attaches one or more solution proposals to an incident to provide the requester with information about how to solve the issue.
solution template	A container that is used to organize items that can be reused in customer-specific solutions.
sort	To rearrange the content of a column in a table in ascending or descending order. The rest of the content in the table is rearranged accordingly.
source language	The language from which text is translated. The source language supported by the studio is English.
static code list	A code list whose values cannot be changed by the customer. The static code list is created in the studio by using the "Create a Code List Data Type" function.
step	A stage in a guided activity towards the completion of a task.
structure element	A characteristic, formula, or restriction in a characteristic structure.
Subject	A short description of an item, such as a task or notification.
subnode	A lower-level node in a hierarchical structure.
subview	A view that a user can open within a view.
system message	A short text that provides information about system feedback to the user. System message texts are displayed in the message region as error messages, warnings, success, or information messages.
tab	A label that the user clicks to access content. Tabs display horizontally in a tab strip and are positioned above the content that they describe.
tab strip	A horizontal row of tabs.
tag	A non-hierarchical keyword or term that is attached to an item. Tags are used to categorize or index items.
target language	The language to which text is translated. The target languages supported by the studio are German (DE), Spanish (ES), French (FR), Portuguese (PT), and Chinese (ZH).

task	A unit of work that has to be performed by a user. Examples include Business Task Management tasks, IT Service and Application Management tasks, and Activity tasks.
task description	Text that provides a brief explanation about a task.
task type	A predefined content type in the system. It has a set of properties such as task category and related business document type which characterize the task. Each task type is related to one task category.
tax decision tree	A function that is created in the studio and used to automatically determine a tax event for a country that does not have an SAP country version in SAP's cloud solution.
Tax Decision Tree Wizard	A guided activity for creating a tax decision tree.
technical user	A back-end process or service that is used to automate technical tasks in the system.
thing	The fundamental component of the current UI design concept and the conceptual entity with which end users work. Things are smart and portable work objects such as a Material or a Business Partner. They embed attributes (data or images), related objects, analytical data, and actions generated from metadata.
thing inspector floorplan	Combination of UI patterns that allows the user to get a complete view of a thing. It shows all aspects of a thing and is a compact summary of all primary attributes, the tags, and the actions that can be carried out.
thing type floorplan	Combination of UI patterns that describes the general UI properties of a thing, such as its name and various icons on the UI, its binding against a business object, and its key.
thing-based navigation	A type of user interface interaction based on business objects defined as "things". This type of navigation is used in SAP's cloud solutions (except for SAP Business ByDesign).
tooltip	A very short text that displays the name of a UI element when the label of the UI element cannot be seen or is only partly displayed to the end user. A tooltip is displayed when the user points the mouse cursor on the UI element.
Toolbox	Tool window in the user interface designer. It provides the panes and controls that can be used to design the layout of a user interface component on the Designer tab.
trace	A trace that is attached to an incident. Incident traces are tied to the life span of the incident.
Trace Explorer	A tool for gathering and analyzing diagnostic information.
trace sequence	The series of recorded method calls and evaluated trace messages that are created as a result of running a trace.

tracing	The process of creating and storing detailed information about runtime events in the system.
translatable text string	An element of a computer readable file that represents a text, such as a field label, a field value, or another user interface text element. A translatable text string can be extracted from a file for translation into a target language.
translated text string	An element of a computer readable file that represents a text, such as a field label, a field value, or another user interface text element, and that has been translated into a target language.
troubleshooting	A set of documents that provide solutions for typical problems or user errors. These documents are available in the advanced search and in the search for incident reporting.
UI Designer	A tool that is part of the studio and that is used to design or enhance the user interface of the solution. You open the UI Designer via the studio.
UI switch	A control in SAP's cloud solution that is used to restrict the visibility of UI changes.
unicode	A 16-bit character set that represents commonly used characters, for example letters and digits, in digital form. Unicode has a distinct advantage over the 8-bit character set ASCII, in that it can render a much larger set of characters. For example, Unicode can represent over 30,000 distinct coded characters, whereas ASCII can only represent 128.
URL mashup	A mashup that sends data from the system to SAP's cloud solution to the Uniform Resource Locator (URL) of an online service provider. The service provider uses the data, for example, to perform a search, and the results are displayed in a new browser window.
user	A business object created for an employee or a service again who needs to log onto the system. It contains technical information such as a password and access rights. Internal components also have user IDs assigned to them.
User and Access Management	A work center view that is used to edit and review user attributes and access rights. The User and Access Management view is a superordinate view of the Business Users view and Support and Technical Users view.
user ID	A unique identifier for an individual who interacts with the services supplied by a system.
user interface dependency	An action that governs the assignment of views to a user.
user interface designer	A tool used to design or enhance the user interface of the solution.
user management	The process of managing all users on the system. This involves applying basic settings such as user names and passwords, assigning users to work centers, and placing restrictions on user access.

validation	A script file that returns either true or false based on a specified condition. Validations are triggered on the execution of events or actions. A validation can raise a message or stop further processing of an event or action.
value selection	The value selected to which a characteristic is restricted. The data shown in a report is restricted to the values selected so that not all data is displayed.
variable	A parameter of a report that restricts a characteristic to one or more specified value selections.
Web service	A Web-based application programming interface (API) that can be used to access external data provided by online service providers. Using mashups, key users, and partners can integrate data provided by Web services with internal data from SAP's cloud solution.
Web service authorization	The process of authorizing users to access a business object by using a service operation.
Web Services Description Language	An XML-based language that is used to describe the function of a Web service and how Web services can be accessed.
Web Services Inspection Language	A specification that supports you in searching for Web services that are available on a specific Web site. As a result the Web services and their endpoints are listed in a Web Service Inspection document.
window	The element of the graphical user interface used for displaying data and applications.
work center	An area where a user can perform tasks or functions that they are responsible for within a company. Examples of work centers are Purchase Requests and Orders and Personnel Administration.
work center proposals	A list of work centers proposed by the system based on the organizational assignment of the user.
work center view	A subdivision of a work center.
worklist	A list of items which may require user action.
XML Localization Interchange File Format	A standard XML-based file format used for exchanging bilingual content between systems during translation.
XML Schema Definition Language	An XML-based language that offers facilities for describing the structure and for constraining the contents of XML documents.

13 Important Disclaimers on Legal Aspects

This document is for informational purposes only. Its content is subject to change without notice, and SAP does not warrant that it is error-free. SAP MAKES NO WARRANTIES, EXPRESS OR IMPLIED, OR OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Coding Samples

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint where to find supplementary documentation. SAP does not warrant the availability and correctness of such supplementary documentation or the ability to serve for a particular purpose. SAP shall not be liable for any damages caused by the use of such documentation unless such damages have been caused by SAP's gross negligence or willful misconduct.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as “sales person” or “working days”) is used. If when referring to members of both sexes, however, the third person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

www.sap.com/contactsap

© 2020 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.

THE BEST RUN SAP