

SAP Add-On Assembly Kit 5.0 Support Package 1.1
Document Version: 1.0 – 2016-05-23

SAP Add-On Assembly Kit



Content

1	SAP Add-On Assembly Kit	5
1.1	Authorization Roles	6
2	About This Document	7
2.1	Naming Conventions	8
2.2	New Features in Version 5.0	9
3	Background Information: Add-Ons	12
4	Software Component Layers	14
4.1	Software Component Hierarchy from SAP Web AS 6.40	16
5	Process Flow: SAP Add-On Assembly Kit	18
6	Defining the Delivery Strategy	19
7	Defining the Namespaces	21
7.1	Rules for Namespaces	22
7.2	Reserving a Namespace	23
7.3	Add-On Software Component	24
7.4	Add-On Release Name	25
7.5	Naming Conventions for Delivery Requests	27
8	Setup of the System Landscape and Systems	29
8.1	System Landscape for Add-On Development	29
	Setting Up a Development Landscape for the First Add-On Release	30
	Setting Up a Development Landscape for Further SAP Releases	31
8.2	System Landscape for Add-On Maintenance	32
	Setting Up Systems for Add-On Maintenance	33
8.3	Developing Multiple Add-Ons in a System Landscape	35
8.4	Client Layout and Transport Paths	36
8.5	Installing SAP Add-On Assembly Kit	38
8.6	Making Settings for the Development and Consolidation System	39
	Entering a Namespace and Defining a Namespace Role	39
	Creating or Updating an Add-On Software Component and Add-On Release Name	40
	Setting the System Change Option	41
	Configuring Parameters for the Transport Control Program tp	42
8.7	Making Settings for the Maintenance Systems	44
8.8	Upgrading the Development and Maintenance Landscape	44

8.9	Test Systems	45
9	Add-On Development	46
9.1	Creating and Assigning Packages (Transaction SE80)	46
9.2	Rules for Add-On Development	46
	Rules for Developing Add-Ons in Multiple SAP Releases	47
9.3	Additional Information About Enhancements and Modifications	48
9.4	Documentation and Translation	50
10	Delivering the Add-On Software	52
10.1	Package Types	53
10.2	Final Assembly System	56
10.3	Creating Deliveries	57
	Object List Checks	61
10.4	Creating Add-On Installation Packages	68
10.5	Making a Delivery Available	68
	Template: Installation Guide	69
10.6	Delivering Languages Translated Retroactively	71
11	Maintenance and Upgrade	75
11.1	Rules for Add-On Maintenance	76
11.2	Creating Maintenance Packages	76
	Creating Add-On Support Packages	76
11.3	Creating Add-On Upgrade Packages (Upgrading the Add-On Software)	77
11.4	Add-On Behavior in SAP System Upgrades	78
11.5	End of Maintenance	80
12	Add-On Uninstallation.	81
12.1	Checklist: Development Aspects.	82
12.2	Checklist: Landscape Aspects.	83
12.3	Checklist: System Aspects.	84
12.4	Checklist: Test Aspects.	85
12.5	Checking Add-Ons for Uninstallability.	85
12.6	Attributes for Uninstallations.	87
12.7	Plug-In Interface for Add-Ons.	88
12.8	Handling Object Types.	89
13	Additional Information	113
13.1	Compatibility of SAP NetWeaver Releases.	113
13.2	Overview: Import Tools	115
13.3	Modifications and Their Consequences	115
	Setting Up a Development Landscape for Modifying Add-On.	116
	Maintenance for a Modifying Add-On.	118

	Rules for Developing a Modifying Add-On	119
	Creating Conflict Resolution Transports	120
13.4	Conflicts	122
	Background Information: Conflict Check	122
	Conflicts When Installing/Upgrading an Add-On	123
	Conflicts When Importing Support Packages	127
13.5	Examples: Attributes in Software Delivery Assembler.	128
	Examples: Attributes for Add-On Uninstallations.	128
	Examples: Attributes for Enhancement Packages.	129
	Examples: Attributes for SAP HANA.	130
13.6	CDs for Add-On Deliveries	130
13.7	Troubleshooting	133
13.8	Links in SAP Help Portal	133
13.9	Terminology	137

1 SAP Add-On Assembly Kit

Purpose

You can use SAP Add-On Assembly Kit to develop industry-specific, country-specific, or enterprise-specific enhancements to the standard SAP system, plus customer and partner projects, while taking advantage of the full range of add-on techniques. SAP Add-On Assembly Kit provides you with a toolset that supports you throughout the entire software lifecycle of your add-on.

Its detailed documentation helps you to verify the quality of your development work in the planning phases and its tools support you when creating your delivery and installing it. SAP Add-On Assembly Kit also provides you with methods for updating and maintaining your software.

Features

Documentation

The SAP Add-On Assembly Kit documentation describes the following steps in the add-on delivery process:

1. Defining the Delivery Strategy

Here you need to answer questions such as:

- What is the underlying SAP release of your development work?
- Does your development work make modifications to the standard SAP system?
- What is your maintenance and upgrade strategy?
- Do you want your add-on to be uninstalleable?

Once you have answered these questions, you have defined a strategy for the delivery of your add-on.

2. Defining the namespaces

When you use a namespace protected by SAP, you ensure that, once delivered, your development objects do not conflict with objects created by other vendors or by SAP. This is the first quality assurance step for your add-on.

3. Setup of the system landscape

The setup of your system landscape for your add-on development is specified by the delivery strategy you choose. In this step, you also install the SAP Add-On Assembly Kit tools in your system landscape.

4. Add-on development

In this step, you develop the software itself in the systems you have configured. This documentation provides you with rules for avoiding errors in the development phase.

5. Delivering the add-on

The SAP Add-On Assembly Kit tools provide you with help when assembling the delivery and when packing the software into importable packages.

6. Maintenance and Upgrade

SAP Add-On Assembly Kit also provides help in the maintenance and upgrade of your add-on. You can use the SAP Add-On Assembly Kit tools to create importable packages that remove any errors in your add-on, as well as to update your add-on or make sure it can run on a higher SAP release.

7. Uninstalling the add-on

SAP Add-On Assembly Kit contains instructions and checks that verify that your add-on can be uninstalled.

Tools

Software Delivery Composer (SDC, transaction `SSDC`) collects all delivery-relevant parts of your development work, checks their consistency, and composes them as a delivery.

Software Delivery Assembler (SDA, transaction `SSDA`) packs the delivery into an importable package format. You can also use SDA to define any import conditions to be respected before the packages can be imported correctly.

Imports made using SAP Add-On Assembly Kit are performed with SAP import tools. The appendix contains a list of these [import tools \[page 115\]](#).

1.1 Authorization Roles

Only users with the relevant authorizations can compose and create add-ons with the SAP Add-On Assembly Kit.

The SAP Add-On Assembly Kit provides various predefined authorization roles, which can be assigned to users depending on their activities and the tools they will be using. For example, you can define whether a user has change rights or display-only rights, and you can grant specific system authorizations (for example, to change RFC destinations) separately. This ensures security in organizations where tasks are divided across employees.

The following table provides an overview of the available roles and their functions:

Table 1: Overview of Roles

Type of Role	Name of Role	Description of Functions
Read Access	<code>SAP_AAK_SDC_DISPLAY</code>	Display-only authorization for all data for composing an add-on or delivery event (SDC)
Read Access	<code>SAP_AAK_SDA_DISPLAY</code>	Display-only authorization for all data for creating an add-on or package (SDA)
Write Access	<code>SAP_AAK_SDC_CHANGE</code>	Change authorization for composing add-ons and delivery events (SDC)
Write Access	<code>SAP_AAK_SDA_CHANGE</code>	Change authorization for creating add-ons and packages i Note This does not include authorization for creating RFC destinations for read access to SDC data. For this, you need the role <code>SAP_AAK_SDA_RFC</code> .
Write Access	<code>SAP_AAK_SDA_RFC</code>	Change authorization for creating and changing RFC destinations. This role is always required if the SDC system is not the same as the SDA system.

2 About This Document

Validity

This document is an overview of the process when using SAP Add-On Assembly Kit 5.0 to create add-ons:

SAP Add-On Assembly Kit 5.0 is available for SAP components based on SAP NetWeaver 7.0 and higher SAP NetWeaver releases.

Note

Note that you can use SAP Add-On Assembly Kit for ABAP development work only.

Target Audience

The information in this documentation is intended for the following groups:

- Add-on producers who are creating add-ons for their own enterprise or for their customers
- System administrators who are setting up systems for add-on development and add-on maintenance
- Software developers involved in add-on development
- Add-on assemblers who are using the SAP Add-On Assembly Kit to create add-on packages

Prerequisites







Before you work with SAP Add-On Assembly Kit, you should have knowledge of the following topics:

- Installation of SAP systems
For further information, see SAP Service Marketplace under <https://service.sap.com/instguides>.
- System copy
For further information, see SAP Community Network under [System Copy and Migration](#).
- Setting up the system landscape (including Change and Transport System)
For further information, see the documentation [Change and Transport System](#) in SAP Help Portal.
- ABAP programming basics
For further information, see the documentation [Application Development on AS ABAP](#) in SAP Help Portal.
- ABAP software maintenance
For further information, see the documentation [Software Logistics](#) in SAP Help Portal.
- Documentation and translation tools
For further information, see the documentation [Services for Information Developers and Translators](#) in SAP Help Portal.

Overview of SAP Notes

The following SAP Notes are important when working with SAP Add-On Assembly Kit and are referenced in this documentation:

Table 2:

Note Number	Short Description
104010 	Restrictions on development in namespaces
105132 	Reserving namespaces from R/3 Release 4.0
16466 	Customer namespace for SAP objects
195442 	Language imports and support packages
212876 	New archiving tool SAPCAR
1883223 	Note on general add-on uninstallations

Integration

This documentation describes the processed required when creating an add-on using SAP Add-On Assembly Kit.

For detailed information about the SAP Add-On Assembly Kit tools, use the help button in the tool in question to display its online documentation

2.1 Naming Conventions

Documentation Links in SAP Help Portal

Any links to further documentation in SAP Help Portal specified here apply to Release SAP NetWeaver 7.4.

Example

[Change and Transport System](#)

To access the documentation in question, select the link or search for the title or release on SAP Help Portal.

In the case of lower releases, an appendix contains a [list of links to documentation in specific releases \[page 133\]](#).

2.2 New Features in Version 5.0

General Information

- You can use Support Package 1 of SAP Add-On Assembly Kit 5.0 to download the PAT file of a package as an SAR file. You can then use the transaction codes SPAM and SAINT to download this SAR file directly to the system. You can also reopen any confirmed deliveries.
- For the first time, Version 5.0 of SAP Add-On Assembly Kit makes it possible to create uninstallable add-ons.
- Version 5.0 of SAP Add-On Assembly Kit supports SAP NetWeaver 7.0 and higher releases.

Software Delivery Composer

- Confirmed deliveries can be reopened
- Object list checks
The following object list checks were added:
 - Can objects be deleted in an uninstallation?
Add-on installations cannot delete every category of data from the system. This check searches the delivery request for any objects and table entries that cannot be deleted. If these objects are delivered, the add-on cannot be uninstalled.
 - Generic checks and automatic corrections
The content of the delivery request is checked for obsolete objects. Any changes to the current object types (for example, replacements by logical transport objects) are made automatically. No further manual modifications are necessary. These checks are always performed.
- A comment function was added in the results list of the object list checks.

Software Delivery Assembler

- The PAT file of a package can be downloaded as an SAR file.
- The SSDA online documentation was updated.
- The following new extended attributes were added:
 - DEINSTALL_ALLOWED
For the package types AOI, AOU, and AOX only: This software component version can be uninstalled. Also see the information about [Add-On Uninstallation \[page 81\]](#). Possible value: A valid note number. Since you cannot create SAP Notes yourself containing information about uninstalling your add-on, the SAP Note [1883223](#) can be used instead. If you do not specify a value for this attribute, the system enters [1883223](#) by default.
 - DEINSTALL_PLUGIN
For the package types AOI, AOU, and AOX only: If this attribute is set, the methods defined in an interface class of the add-on in question are called in the uninstallation process. This is applicable if the following prerequisites are met:
 - This software component version can only be uninstalled under certain conditions. This decision can be made in the local system only.

- The add-on in question created dynamic objects. To delete the add-on in the uninstallation process, a method of the interface class of the add-on is called. This method returns a list of the dynamically created objects to the uninstallation framework as a return parameter and hence makes it possible to delete these objects.

Also see the information under [Attributes for Uninstallations \[page 87\]](#). You use the software component version to deliver the class of the add-on in question. Specify the name of the calling interface class in the add-on as the value for the attribute.

- **KERNEL_VERSION**

From SAP Web AS 6.10: Specifies the kernel version required to import the package, for example: 620/00123.

- **LANGUAGE_BY_SP**

This attribute can be used for packages of the type AOU and CSP from SAP NetWeaver 7.0. If new languages are delivered with a packages with one of these types, they can be registered as new installed languages in the import to a target system by setting the attribute LANGUAGE_BY_SP. The languages specified in the attribute must already exist in the attribute LANGUAGE. The values for this attribute must be specified using the two-character ISO language key.

Example

LANGUAGE_BY_SP ISO-FRJA

This package delivers the languages FR (French) and JA (Japanese).

- **NEEDED_DBSYS**

Applies to packages of the type AOI, AOU, and AOX from SAP NetWeaver 7.0. Specifies a list of database systems where the add-on can be installed.

A warning appears if the package is imported on any other database system. Despite this, the package can still be imported, but cannot run on this system. This step may be necessary, for example, before a database migration.

If you want the package to only be imported on one of the specified database systems, add **:R** to the list. **:R** can only be specified at the end of the package list. In this case, the system refuses to install the package on any other database system than the ones specified.

The current possible values are:

- ADA = MaxDB
- DB2 = IBM DB2/390
- AS4 = IBM DB2/400
- DB6 = IBM DB2/LUW
- HDB = HANA
- INF = Informix
- ORA = Oracle
- MSS = MS SQL Server
- SYB = Sybase ASE

Example

The package is to be imported on one of the following database systems: NEEDED_DBSYS
HDB,INF

The package must be imported on one of the following two database systems: NEEDED_DBSYS
HDB,INF:R

- R3TRANS_VERSION

Specifies the minimum version of the transport program R3trans required to import the package. Specify the version as a time stamp: **<year, 4 characters> <month, 2 characters><day, 2- characters>**. You can omit any trailing zeros.



Example

20061025

- TP_VERSION

Specifies the minimum version of the transport program `tp` required to import the package. Specify the version in the following format: **<KERNEL_VERSION>/<VERSION>**.

- UPG_KEY_REQUEST

For the package types AOU to SAP Web AS 6.40, AOX from SAP NetWeaver 7.0, and AOS, included in the SAP upgrade: The SAP upgrade tool prompts the user to enter a key before including the package in the upgrade. You cannot create this key yourself. You can, however, request SAP to create it for your add-on package by opening a customer message on component XX-PROJ-AAK. Once SAP has provided you with the key, notify your customers of it, plus any other important information about the upgrade. The attribute UPG_KEY_REQUEST must be used in combination with the attribute SEE_PNOTE.

Possible value: **T** = True

- Post-delivery of modified attributes using attribute change packages (ACP)

The package type attribute change package (ACP) is only used to deliver modified attributes. It is often necessary to add new import attributes or correct existing attributes after the delivery of OCS packages (such as installation packages, upgrade packages, or support packages). For example, you may need to add new import prerequisites to validate additional system states as a basis for imports. Once the attributes of a released package have been modified using the option *Post-Delivery (with ACP)*, the system creates one ACP for each software component version. If the attributes of multiple released packages in the same software component version are modified, the system creates one ACP version for each package.

- Modified tab order for package registration
- Modified display and editing of the attributes and import conditions
- New roles introduced

Documentation

- The new and updated functions in SAP Add-On Assembly Kit 5.0 have been included in the documentation.
- Information about SAP NetWeaver 7.4 has been added, for example paths to other documentation have been replaced by links and the links changed to reflect the supported releases.

3 Background Information: Add-Ons

An add-on consists of a series of transport requests grouped together as a delivery package. When imported, the add-on is registered as an additional software component in the system.

Add-ons are developed in separate systems based on a specific SAP release. The functions of an add-on are based on the functions of main SAP components or SAP application components.

An add-on can contain the following objects:

- New add-on objects
- Add-on-specific customizing
- Modified objects from main SAP components or SAP application components (if they adapt certain functions to the requirements of the add-on)

Add-ons must not modify SAP kernel objects or SAP Basis objects and are independent of the operating system.

Add-ons are installed in SAP systems by SAP Add-on Installation Tool (transaction SAINT). The way add-ons are installed is specified by the SAP release and potentially also by the support package level on which you developed the add-on. This also applies to add-on upgrades.

Add-On Package Types

You can use SAP Add-On Assembly Kit to create the following [package types \[page 53\]](#):

- Add-on installation package (AOI)
- Add-on upgrade package (AOU)
- Add-on support package (CSP from SAP NetWeaver 7.0 (AOP up to SAP Web AS 6.20))
- Add-on exchange package (AOX) from SAP NetWeaver 7.0
- Conflict resolution transport (CRT)
- Attribute change package (ACP)

Benefits of Delivering Software as Add-Ons

You can deliver software that expands the standard SAP system in different ways:

- As transport requests

This has the following drawbacks:

- The transports are not imported in a defined order. This can cause import errors, downgrades, and loss of data.
- Import prerequisites are not checked. There is a risk that transports are imported into systems with an incompatible software component state. This can also cause import errors, downgrades, and loss of data.
- Conflicts with other software components are not detected. This can cause errors in other SAP applications.

-
- Conflicts with customer modifications are not detected. Customer modifications can be overwritten without warning.
 - The exported transport objects are not made anonymous.
 - The content of transports is not respected correctly when the add-on or customer system is upgraded. This can cause downgrades and loss of data.
 - As an add-on
Unlike transport requests, delivering software as add-ons has the following benefits:
 - Installation prerequisites can be specified.
 - An installation order can be specified.
 - Conflicts with other software components can be resolved.
 - Conflicts with customer modifications can be resolved.
 - The installation process and upgrade process are reliable.
 - Objects are made anonymous.
 - The delivered software state is visible.

4 Software Component Layers

Software Component Layers to SAP Web AS 6.10

Up to SAP Web AS 6.10, software components in SAP systems are in one of two layers:

- Layer of the main components (such as SAP_BASIS, SAP_ABA, and SAP_HR)
- Layer of the add-ons (such as plug-ins, industry solutions, development projects, customer add-ons, and partner add-ons)

This means that all types of add-ons are handled in the same way. It is not possible to define dependencies within the add-on layer and stack add-ons on each other.

CVERS



Figure 1: Software Component Layers to SAP Web AS 6.10

Software Component Hierarchy from SAP Web AS 6.20

To make it easier to handle add-ons, SAP implemented an extended software component hierarchy in SAP R/3 Enterprise (based on SAP Web AS 6.20). This hierarchy covers two layers of mandatory (fixed) software components and four layers of optional software components. Mandatory software components can only be installed in an SAP system using a system installation or an SAP upgrade. Optional software components, on the other hand, can be installed retroactively.

In the software component hierarchy, references from add-ons can only be made from a higher layer to the layer below (for example, from layer C to layer I).

Add-ons created using SAP Add-On Assembly Kit are always in layer C.

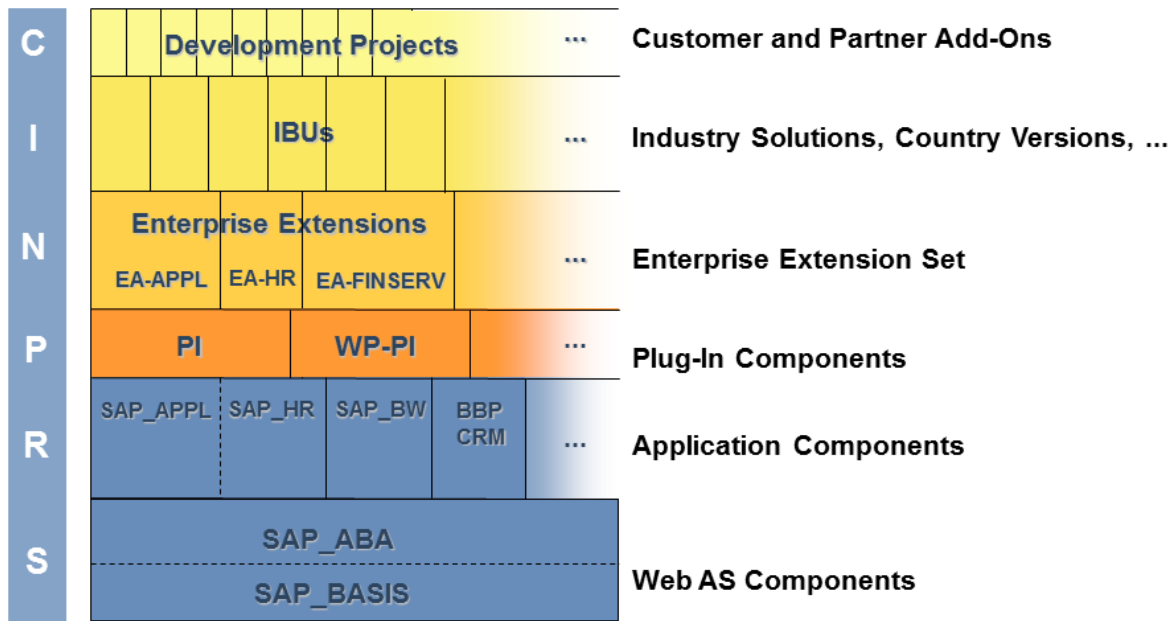


Figure 2: Layers of the Software Component Hierarchy in SAP Web AS 6.20

The software component hierarchy consists of the following layers.

Mandatory Software Components (Main Components)

- *Web AS components (layer S)*
These include the SAP Web Application Server components: SAP_BASIS and SAP_ABA.
- *Application components (layer R)*
These include all SAP system software components that cannot be installed retroactively and that are not part of SAP Web AS, for example SAP_APPL, SAP_HR, SAP_APO, and SAP_BW. These components can only be installed or removed from a system in an upgrade.

Optional Software Components (Add-On Components)

- *Plug-in components (layer P)*
Plug-in components are add-ons that provide functions for a wide range of other components and are not specific to one application. A typical example is the plug-in PI.
- *Enterprise Extensions (layer N)*
These include add-ons that expand the core functions of an SAP system. Their main aim is to delivery application component functions in a faster cycle than is possible in a full system release.
- *Industry solutions and country version (layer I)*
These include add-ons used to adapt the application components plus any plug-ins and enterprise add-ons installed to the requirements of specific industries or countries. They can also include additional functions required by only a small number of customers.
- *Development projects (layer C)*
These include all add-ons that are not part of one of the other layers, such as customer add-ons and partner add-ons.

4.1 Software Component Hierarchy from SAP Web AS 6.40

The software component hierarchy with six layers introduced in SAP Web AS 6.20 was expanded and modified again in SAP Web AS 6.40. The Basis plug-in component (PI_BASIS) became a part of SAP NetWeaver and is in its own layer. The application components are based on this layer. SAP_BW is now also a part of SAP NetWeaver. The layer for industry solutions and country versions was expanded to multiple stacked layers. This made it easier to map dependencies between industry solutions or country versions.

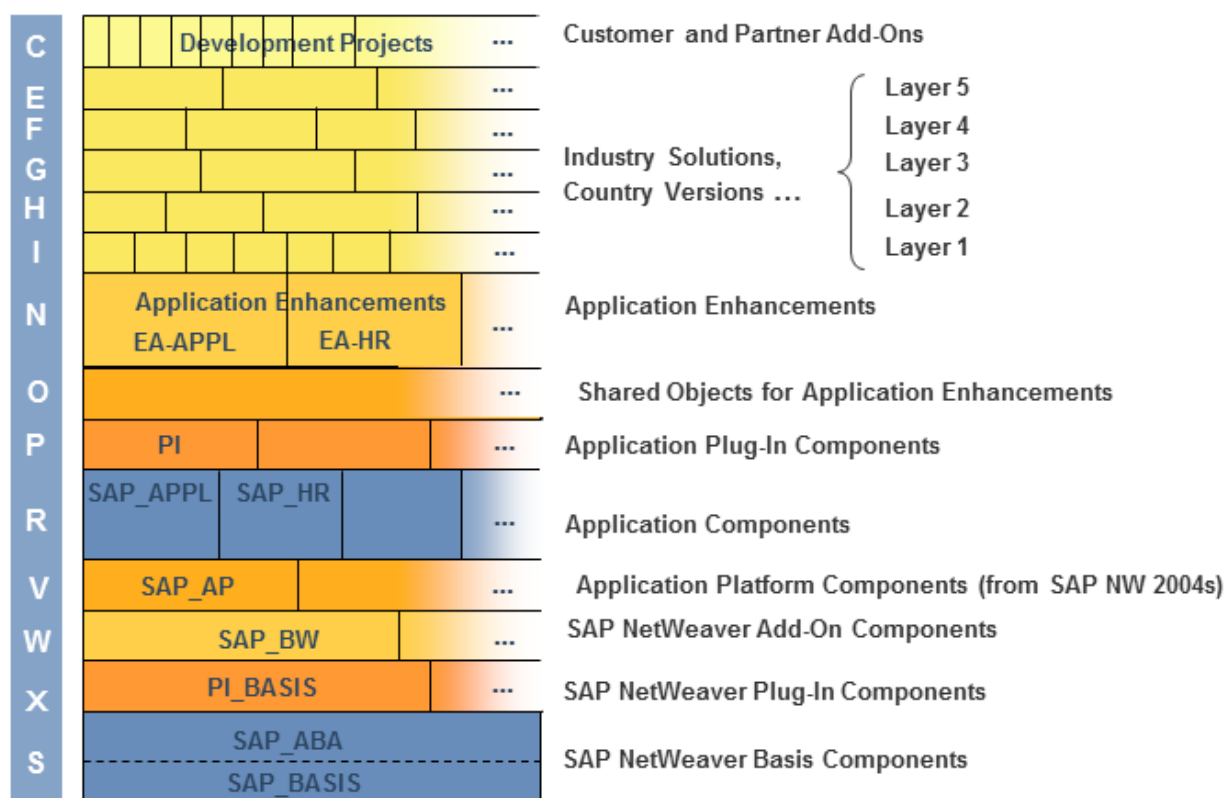


Figure 3: Layers of the Software Component Hierarchy for SAP NetWeaver

The software component hierarchy consists of the following layers.

SAP NetWeaver Software Components

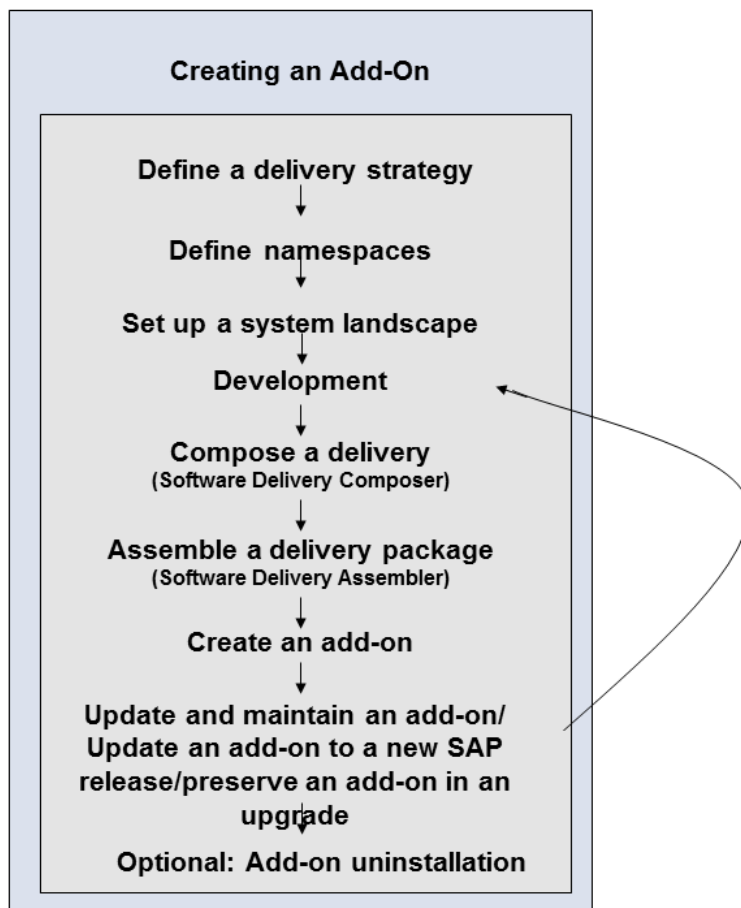
- *NetWeaver Basis components (layer S)*
These are the components SAP_BASIS and SAP_ABA.
- *NetWeaver plug-in components (layer X)*
These are the Basis plug-in components in SAP NetWeaver. A typical example is PI_BASIS.
- *NetWeaver add-on components (layer W)*
These are additional add-on components in SAP NetWeaver- A typical example is SAP_BW.
- *NetWeaver application platform components (layer V)*
(From SAP NetWeaver 7.0)
These are the components of the NetWeaver application platform, such as SAP_AP.

Further Software Components Based on SAP NetWeaver

-
- *Application components (layer R)*
These are software components that are not part of the SAP NetWeaver platform and that include the core functions of the SAP ECC system, such as SAP_APPL or SAP_HR. These components cannot be installed in an SAP system retroactively and can only be updated by an SAP system upgrade.
 - *Application plug-in components (layer P)*
These are add-ons that provide functions for a wide range of other components and are not specific to one application. A typical example is the plug-in PI.
 - *Application enhancements layer N)*
These are add-ons that contain or expand the core functions of an SAP system.
 - *Industry solutions and country versions (layers E, F, G, H, and I)*
These are add-ons containing additional functions or used to adapt the application components plus any plug-ins and application enhancements installed to meet the requirements of specific industries or countries. These add-ons are assigned to one of the five layers to make them stackable.
 - *Development projects (layer C)*
These include all add-ons that are not part of one of the other layers, such as customer add-ons and partner add-ons. Add-ons created using SAP Add-On Assembly Kit are always in layer C.

5 Process Flow: SAP Add-On Assembly Kit

The following figure is an overview of the process flow when using SAP Add-On Assembly Kit to create add-ons:



1. [Defining the Delivery Strategy \[page 19\]](#)
2. [Defining the Namespaces \[page 21\]](#) (for development objects and the add-on software component)
3. [Setup of the System Landscape and Systems \[page 29\]](#)
4. [Add-On Development \[page 46\]](#)
5. [Delivering the Add-On Software \[page 52\]](#)
6. [Maintenance and Upgrade \[page 75\]](#)
7. [Add-On Uninstallation \[page 81\]](#)

6 Defining the Delivery Strategy

By answering the following questions before you set up the system landscape, you can define your **delivery strategy**

- In which SAP release does development take place?
Define the SAP release in which you want your development work to take place. If the add-on has dependencies on databases, define the database too. The operating system is not relevant here, since developments are not allowed to be platform-specific.
- How are further SAP releases supported?
We recommend that add-ons are developed for as many releases as possible. This means that the code of the developed objects must run on all supported SAP releases. It must not use syntax that is no longer supported in later releases. You require a development landscape for each supported SAP release, but you can transport your development to the releases without any problems.
- Are modifications to the standard SAP system required?
If your add-on objects modify the SAP system and these modifications are also delivered, the add-on is a modifying add-on. Modifications have consequences throughout the life cycle of the add-on. It is essential that you read the information about consequences of making modifications in the sections [Modifications and Their Consequences \[page 115\]](#) and [Conflicts \[page 122\]](#).

➔ Recommendation

We recommend that you only create **non-modifying add-ons**. SAP does not provide certification for add-ons that contain modifications.

- What is the maintenance strategy for the customer delivery?
If you want to develop a modifying add-on, you are dependent on the maintenance cycle of the modified component. You must react to every SAP support package that has conflicts with the add-on. Non-modifying add-ons are not dependent on the SAP maintenance cycle and you can create support packages whenever you need them. In this case too, however, you should define a delivery cycle for your support packages.
- What is your upgrade strategy?
You must define an upgrade strategy for your add-on before starting development of the add-on. You must define in which intervals you want to update your add-on and the behavior of your add-on in SAP system upgrades. Note that, because your add-on must always react to a new release, it is always dependent on the SAP release cycle
Even if you do not want to update your add-on in an SAP system upgrade, you must ensure that it is preserved and continues to work in the new release. This requires you to run tests for your add-on for each new SAP release. If you do not support a particular SAP release, your add-on customers cannot upgrade to this release. For more information, see the sections [Add-On Behavior in SAP System Upgrades \[page 78\]](#) and [End of Maintenance \[page 80\]](#).
- Do dependencies on other add-ons exist?
If your add-ons are based on other existing add-ons, you must first define dependencies. This is the case, for example, if you are developing an add-on that has another add-on as a prerequisite. You must describe this prerequisite relationship.
- Does your add-on contain customizing?

SAP Add-On Assembly Kit only supports deliveries of customizing settings using **BC Sets**. This means that you must define a development and delivery strategy for your customizing settings in addition to your add-on delivery strategy.

- In which languages do you want to deliver your add-on?
You must define the languages you require for your add-on. If you require more than one language, you must organize translation.
- Do you want your add-on to be uninstalleable?
Before development starts, you must decide whether you want your add-on to be uninstalleable.

7 Defining the Namespaces

Use

There are two traditional name ranges in SAP systems: The SAP name range for development at SAP and the customer name range for customer developments.

Naming conflicts can occur, however, when add-ons are delivered by add-on producers.

Example

For example, a company develops enhancements to standard SAP applications and delivers them to third-party companies that potentially make their own developments. If the add-on producer and consumer both work in the same customer name range, some objects may potentially be given the same name. This causes naming conflicts and needs to be avoided.

The recommended solution is to develop objects in a separate namespace reserved exclusively for an SAP customer or SAP partner. One advantage of developing objects in reserved namespaces is that namespaces are checked against a license key in the namespace table. This prevents the namespace from being used illegally. You can request namespaces in SAP Support Portal.

Another benefit of reserved namespaces is that all add-on objects developed are indicated by an appropriate prefix and cannot conflict with other objects with the same name.

You require the following namespaces when developing and delivering an add-on:

- Namespace for the development objects
Guarantees that the names of the development objects are unique.
- Namespace for the add-on software component
Guarantees that the add-on software component is unique
The name of the add-on software component is derived from the namespace reserved for the add-on software component. The names of the delivery requests (and hence the importable packages) are themselves derived from the add-on software component.

Note

The namespace for the add-on software component can be the same as the namespace for the development objects. If you are only developing a single add-on, for example, and you have already reserved a namespace for the development objects, you can also use this namespace for the add-on software component. Note also that the name of the add-on software component is visible in the system and is reflected in the name of the importable add-on packages. For this reason, you must decide which name you want to use when you ship your packages.

Before you start your development work, you must also define the add-on release name, since an add-on is always created in a specific release.

More Information

- [Rules for Namespaces \[page 22\]](#)
- [Reserving a Namespace \[page 23\]](#)
- [Add-On Software Component \[page 24\]](#)
- [Add-On Release Name \[page 25\]](#)
- [Naming Convention for Delivery Requests \[page 27\]](#)

7.1 Rules for Namespaces

When defining namespaces, be sure to observe the following rules:

General Rules

- Define your namespace between two forward slashes. Your namespace can have between three and eight characters.

➔ Recommendation

We recommend that you choose a namespace with at least four characters, since three-character namespaces are assigned internally at SAP.

Example

/ABCD/

- Do not use a namespace that has already been reserved for SAP developments:
 - `/*SAP*/`
 - `/IS*/`
 - `/C<two characters>/`
 - `/HR<two characters>/`
 - `/P<two characters>/`
- Use capital letters.

Additional Information for Namespaces for the Add-On Software Component

To ensure that the [add-on software component \[page 24\]](#) is unique, its name is derived automatically from the namespace reserved for it. When a delivery is created, the namespace prefix in the delivery name is taken over as the add-on software component.

Example

If the namespace for the add-on software component is **/ABCD/** , then your add-on software component must be **ABCD** .

As the name of the add-on software component is visible in the system and is also reflected in the name of the package, you should consider reserving a specific namespace - by which your add-on will then be known - for the add-on software component

Alternatively, you can use the namespace for development objects for the add-on software component. If you have reserved multiple namespaces, you can specify that the name of your add-on software component will be derived from any one of these namespaces.

Example


If you have reserved the namespaces **/ABCD1/** , **/ABCD2/** and **/ABCD3/** , your add-on software component could be called **ABCD2** .

7.2 Reserving a Namespace

Use

You must reserve the following namespaces for your add-on:

- Namespace for the development objects
- Namespace for the add-on software component, if not the same as the namespace for the development objects

You can reserve namespaces in SAP Support Portal under <https://support.sap.com/namespaces>. The namespaces are reserved for you within a few days. You can then release these namespaces by entering them for use in the SAP system. You are sent the required license keys for your namespace when you register the namespace for your installation number.

Prerequisite

Before you can reserve namespaces, your customer number must have an ABAP development license with authorization to develop your own software.

Procedure

The functions discussed below can be found in SAP Support Portal under <https://support.sap.com/namespaces>.

Reserving a namespace

To do this, choose [Request namespace](#) in the namespace application.

When you choose your namespace, note the [rules for namespaces \[page 22\]](#) that apply to SAP Add-On Assembly Kit.

You can track the current status of your namespace request under [Status of requests](#).

Generating a key for the namespace

To do this, choose [Generate key](#) in the namespace application.

Before you can do this, SAP must confirm the reservation of the namespace and you must specify your installation number when you reserve the namespace. Any installation numbers not specified in the reservation can be entered as valid installations for the namespace later using ► [Change namespace](#) ► [External assignment](#) ►. This means you can then the namespace license keys for these installation numbers too.

Note

To [create deliveries \[page 57\]](#), you require the development license for the installation number of the consolidation system.

For a detailed description of the functions in question (and the authorizations required), see the documentation in SAP Support Portal under <https://support.sap.com/namespaces> or in SAP Note [105132](#).

Remember that you must [enter the namespace and define the namespace role \[page 39\]](#) after the reservation.

7.3 Add-On Software Component

A software component bundles a set of packages (development classes) that can only be delivered to customers together. All packages are distributed disjunctively among software components. This means that the objects in a package can only be delivered to customers with a software component. You assign objects to a software component by assigning the package to which the objects belong. In general, there are several releases for each software component.

You need the add-on software component in the following situations:

- For assigning packages to the software component
- For creating the delivery and delivery requests with Software Delivery Composer
- For generating importable packages with Software Delivery Assembler
- For importing packages with Support Package Manager or SAP Add-On Installation Tool

To ensure that the add-on software component is unique, its name is derived automatically from the namespace reserved for it. When a delivery is created, the namespace prefix of the delivery name is taken over as the add-on software component.

7.4 Add-On Release Name

In most cases, multiple releases exist for a single software component. The add-on release name provides your add-on development work with a structure.

Note the following points when defining the add-on release name:

- Use only numerals, uppercase letters, and underscores and do not use periods or other special characters.
- Your release name must have at least three characters and have no more than 10 characters.
- The release name must be part of an ascending sequence (both in ASCII and in EBCDIC).
- Use a format that can ensure this sequence and also provides correct results when release names are compared.

Example

100_700 < 200_700 < 300_700 < 400_700

- The add-on release name is reflected in the version name of the [delivery request \[page 27\]](#) and is restricted to three places here, which means you should choose an add-on release name from which a unique version name for the request can be derived.

Examples

If you define the add-on release name as recommended in these examples, you also ensure that a unique version name of the delivery request can be derived from the release name. For your add-on, select the case that best suits your delivery strategy.

- Your add-on release supports multiple SAP releases.
If your add-on supports multiple SAP releases at the same time (for example, add-on release 100 supports SAP NetWeaver 7.0 and higher), you can use the following syntax for the add-on release name:

<Ziffer>00_<zugrunde liegendes SAP-Release>

The first numeral is free, numerals two and three are both 0, and the fourth place is a separator dash. The subsequent numerals reflect the underlying SAP release.

The version name of the delivery request uses the first numeral of the add-on release name and the first two places in the underlying SAP release,

Example

Table 3:

SAP release name	Add-on release name	Version name of the delivery request
700	100_700	170
700	200_700	270
731	200_731	273
740	300_740	374

- You create a new add-on release each year.

If you publish precisely one add-on release each year, you can use the following syntax for the add-on release name:

`<Ziffer>00_<Erstellungsjahr des Pakets>`

The first numeral is free, numerals two and three are both 0, and the fourth place is a separator dash. The subsequent numerals reflect the year in which the package was created.

The version name of the delivery request uses the first numeral of the add-on release name and two unique places in the year in which the add-on release was created.

Example

Table 4:

Add-on release name	Version name of the delivery request
100_2014	114
200_2014	214
200_2015	215
300_2016	316

- Your add-on release supports precisely one SAP release.

If a release of your add-on supports precisely one SAP release (for example, add-on release 100 supports SAP NetWeaver 7.0, add-on release 200 supports SAP NetWeaver 7.31, and add-on release 300 supports SAP NetWeaver 7.4), you can use the following syntax for the add-on release name:

`<Ziffer><Ziffer>0`

The first two numerals are free and the third numeral is 0. The further places are not used.

The version name of the delivery request matches the add-on release name.

Example

Table 5:

Add-on release name	Version name of the delivery request
100	100
110	110
200	200
300	300
450	450

Note that you must always develop a new add-on release on a higher SAP release and ensure that your release names ascend in sequence. If not, no add-on exchange upgrades can be performed.

i Note

Before you start development work, [enter the add-on software component and the add-on release name \[page 40\]](#).

7.5 Naming Conventions for Delivery Requests

The name of the delivery requests is derived from the add-on software component and the add-on release name and guarantees that the delivery requests and the importable add-on packages are unique.

The name of the delivery requests must use the following syntax:

SAPK-<version name><++>IN<namespace>

Table 6:

Part of Name	Description
SAPK-	Prefix string
<version name>	<p>Three-character add-on software component release</p> <p>When the delivery requests are created, Software Delivery Composer uses the first three characters of the add-on release name, for example 100, as a default for the version name.</p> <p>You can replace this default with a name that suits your delivery requests. For examples of how to define the version name, see Add-On Release Name [page 25].</p>
<++>	<p>Two characters: The characters 0-9 and A-Z are supported.</p> <p>We recommend that your chosen characters reflect the type of the delivery request, for example:</p> <ul style="list-style-type: none">• CH: CHange piece list• CO: COmponent piece list• EX: EXchange component piece list <p>If support packages and CRTs are consecutive, the two characters should also be updated consecutively. They should reflect the support package level, for example 01, 02, 03,..., 99, A1, A2, ..., B1, B2,..., Z9.</p>
IN	Separator

Part of Name	Description
<namespace>	Namespace name that matches the name of the add-on software component, for example ABCD. The namespace is taken automatically from the namespace prefix of the delivery name.

Example

If /ABCD/ was reserved as a namespace, the component piece list for add-on ABCD 100_700 is as follows:

SAPK-170COINABCD.

If /ABCD/ was reserved as a namespace, the support package 03 for add-on ABCD 200_740 is as follows:

SAPK-27403INABCD.

8 Setup of the System Landscape and Systems

For information about the setup of a system landscape, see the [Change and Transport System](#) documentation in SAP Help Portal under:

- Transport Management System
- Change and Transport System - Overview

i Note

In your system landscape, verify that the upgrade and maintenance strategy in the development system is compatible with the [delivery strategy \[page 19\]](#).

You required **two systems** for each development level and maintenance level. The consolidation system is also used as the final assembly and translation system.

You also need a temporary test system for final assembly tests. System copies are created to update this test system to the delivered version.

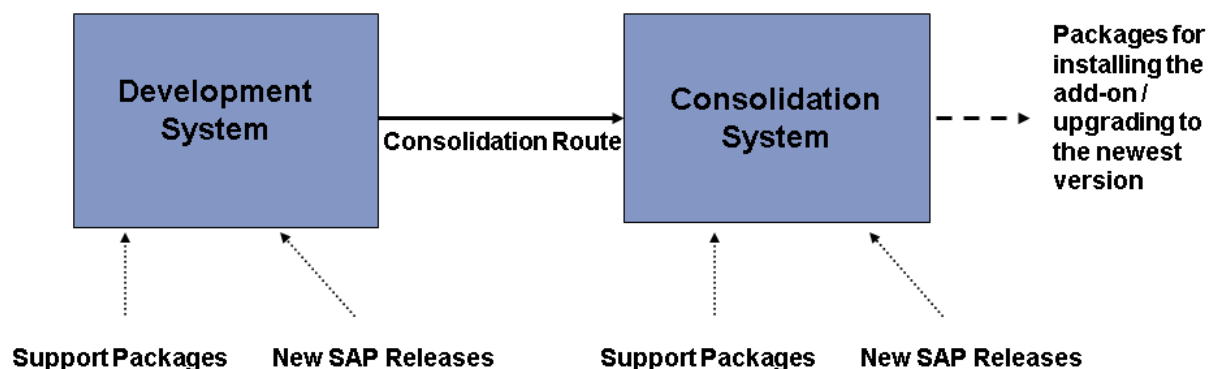
In all system landscapes, the consolidation system is used as the final assembly system and cannot contain any cross-client test data that could accidentally be included in the delivery. You can of course work with client-specific data in test clients.

8.1 System Landscape for Add-On Development

For each add-on release and each supported SAP release, you require two system for add-on development:

- A development system (for software development and documentation and customizing)
- A consolidation system (for translation and delivery. **Do not** make repairs or customizing settings here.)

Create a consolidation route from the development system to the consolidation system and schedule regular consolidation transports.



Note

The consolidation system is also used as the **final assembly system**.

You also require a temporary **additional test system**. For more information, see [Test Systems \[page 45\]](#).

Caution

If you test the add-on functions or customizing changes in the consolidation system directly and not in a test system, you risk including test data in deliveries (if the wrong client is configured).

8.1.1 Setting Up a Development Landscape for the First Add-On Release

Prerequisite

You have defined your system landscape for add-on development and you want to set up the systems for the first add-on release.

Procedure

You can set up the systems for add-on development as follows:

- By installing an SAP system
- By copying an installed SAP system

Note

In both cases, you must reconstruct the state you require for your [delivery strategy \[page 19\]](#), in particular the release state and support package state, when you are setting up the SAP system.

When you are setting up the system landscape, also take care when creating the [client layout \[page 36\]](#).

Result

You have set up the systems. You can now [install SAP Add-On Assembly Kit in your system landscape \[page 38\]](#). To configure the landscape for SAP Add-On Assembly Kit, read the section [Configuring the Development System and Consolidation System \[page 39\]](#).

8.1.2 Setting Up a Development Landscape for Further SAP Releases

Use

If you want your add-on to support multiple SAP releases, you require a separate development landscape for each release in question.

The procedure for setting up a landscape for developing add-ons is described under [Setting Up a Development Landscape for Modifying Add-Ons \[page 116\]](#).

You can set up the systems for non-modifying add-on development work on a different SAP release as follows:

Setting up a development landscape for non-modifying add-ons:

1. Set up the systems in the new SAP release by installing an SAP system or copying an installed SAP system.
2. [Install SAP Add-On Assembly Kit \[page 38\]](#) in the systems and make the required [settings in the systems \[page 39\]](#).
3. If you want to deliver your add-on in multiple languages, set up the translation environment in the new consolidation system in the same way as in the existing consolidation system. You can find more information in [Setting Up and Coordinating Translation](#) in SAP Help Portal.
4. In the delivery client of the consolidation system of the first development landscape, create a transport request by including the component piece list of the predecessor release. Choose [Transport of copies](#) as the request type.
5. Export the transport of copies in the translated languages from the consolidation system of the first development landscape.
6. Import the transport of copies into the customizing client of the new development system and into the delivery client of the new consolidation system.
7. Register the transport of copies as a component piece list of the predecessor release. This piece list is required later when the exchange component piece list is created.
To do this, start Software Delivery Composer in the delivery client of the new consolidation system. In the initial screen, choose ► [Delivery for Delivery Request](#) ► [Register](#) ►.
8. Instruct your developers to verify in the new development system which of the imported add-on objects need to be modified or deleted in the new SAP release. Once all adjustments have been made, transport all add-on objects (including the deleted objects) to the new consolidation system in a consolidation transport. This transport becomes the basis of your new supported SAP release.

Example

The non-modifying add-on ABCD, Release 100, was developed on SAP NetWeaver 7.0 (ABCD 100_700). The same add-on release, 100, now also supports SAP NetWeaver 7.4 (ABCD 100_740). The figure illustrates the setup of the further development landscape.

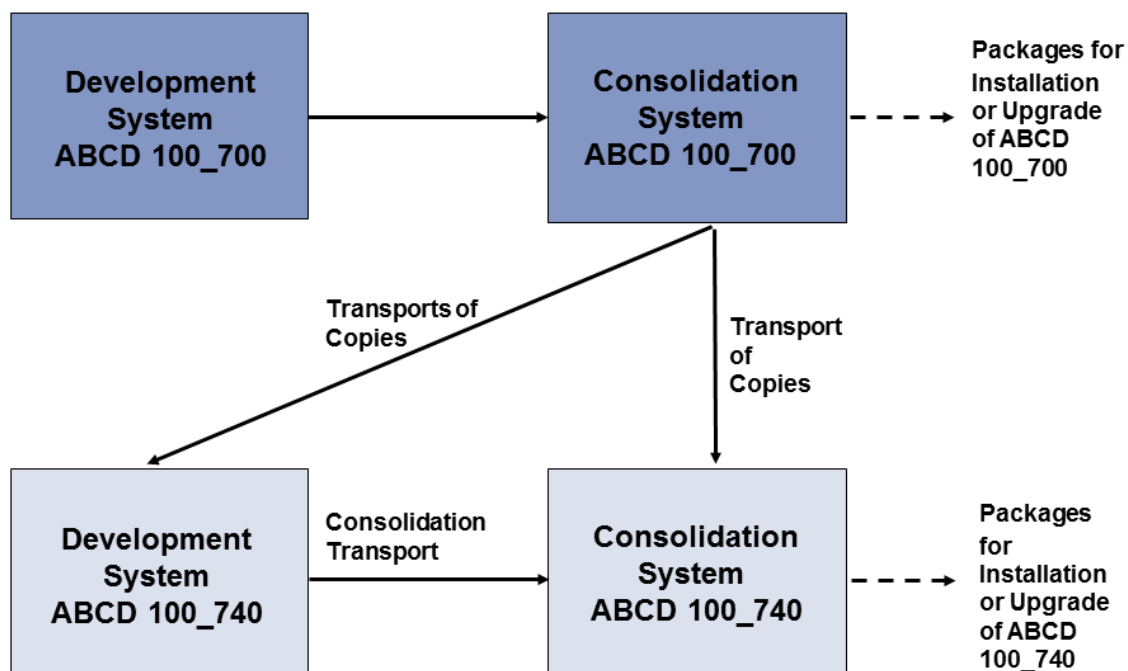


Figure 4: Setup of the Further Development Landscape for the Non-Modifying Add-On ABCD 100_740

Result

You have set up the systems for add-on development work on a further SAP release.

If you want your add-on to support further SAP releases, repeat the procedure above for each SAP release in question.

8.2 System Landscape for Add-On Maintenance

Maintenance involves the following tasks:

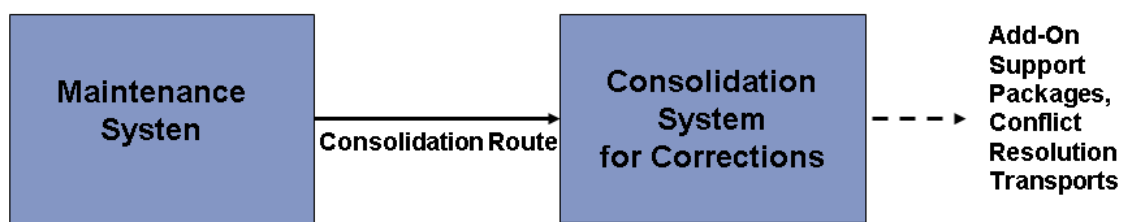
1. Removing any errors in the add-on (which means creating add-on support packages)
2. Creating deliveries for the packages mentioned above

You require two systems for each delivered add-on release that you want to maintain. In a two-system landscape, you can make regular corrections in the maintenance system and reproduce any error situations reported by your customers in the consolidation system. This requires a test client in which cross-client changes cannot be made.

If modifications exist in your system, also read the section [Maintenance Landscape for Modifying Add-Ons \[page 118\]](#).

Two-System Maintenance Landscape

Create a consolidation route from the maintenance system to the consolidation system for corrections and schedule regular consolidation transports.



i Note

The consolidation system is used as both a translation system and as a final assembly system.

Configure the maintenance system in the same way as the development system (see also: [Configuring the Maintenance Systems \[page 44\]](#)) and configure the consolidation system for corrections in the same way as the consolidation system for development.

8.2.1 Setting Up Systems for Add-On Maintenance

Prerequisite

You have defined the system landscape for maintenance of the add-on release in question and want to set up the systems for add-on maintenance.

Procedure

You can set up the systems for add-on maintenance as follows:

- By copying the consolidation system from the development landscape of the add-on release in question

➔ Recommendation

We recommend the system copy as the default method.

- By transporting copies of the objects of the add-on release in question

Copying Your Consolidation System

1. Make sure that all deliveries in the system you want to copy have the status [Confirmed](#).
2. Make a system copy of your consolidation system (which you used to create the add-on release in question) for both the maintenance system and the consolidation system for corrections.

Note

For more information about system copies, see SAP Community Network under [System Copy and Migration](#).

3. Choose one of the following options for the client layout:
 - After making the system copy, use the test client as the correction client and the delivery client as the customizing client in the maintenance system. The client layout of the consolidation system for corrections is preserved after the system copy. (See also [Client Layout \[page 36\]](#).)
 - If you want to base your maintenance on the standard customizing delivery, you can also create a new correction client as a copy of the customizing/delivery client.
To do this, create the correction client as a client copy of the delivery client (of the previously copied consolidation system) in the maintenance system. The former delivery client then becomes the customizing client. In the same way, create the test client as a client copy of the delivery client in the consolidation system for corrections, to ensure that any later tests use the delivery customizing settings.

Transports of Copies

1. Install two SAP system with the release and support package level of the consolidation system when the add-on release in question was last exported.
2. Using the unmodified export state, create a transport request in the delivery client (see [Client Layout \[page 36\]](#)) of the consolidation system and include the component piece list of the add-on release in question in this request. Choose [Transport of copies](#) as the request type.
3. Before you release the transport, verify that the tp parameter `r3transoptions` is set to the value `smodi=yes` in Transport Management System, to ensure that the modification information is transported.
4. Release this transport.
5. In the maintenance system, create the customizing client as a client copy of client 000 and create the delivery client in the consolidation system for corrections in the same way.

Note

Note the information under [Client Layout \[page 36\]](#) for your maintenance systems.

6. Import the transport of copies into the customizing client of the maintenance system and also into the delivery client of the associated consolidation system for corrections.
7. To create further clients, choose one of the following options:
 - In the maintenance system, create the correction client as a client copy of client 000. In the consolidation system for corrections, create the test client as a client copy of client 000.
 - If you want to base your maintenance on the standard customizing delivery, you can also create a new correction client as a copy of the customizing/delivery client.

To do this, create the correction client as a client copy of the customizing client 000 in the maintenance system. Create the test client as a client copy of the delivery client in the consolidation system for corrections, to ensure that any later tests use the delivery customizing settings.

8. Make the [settings in the maintenance systems \[page 44\]](#).
9. In the delivery client of the consolidation system for corrections, register the transport of copies as a component piece list of the add-on release in question. This is needed for the object list check [New Objects \(Entries\) in Support Package/CRT](#) and for conflict checks when creating conflict resolution transports. To do this, start Software Delivery Composer in the delivery client of the consolidation system for corrections. In the initial screen, choose ► [Delivery for Delivery Request](#) ► [Register](#) .

8.3 Developing Multiple Add-Ons in a System Landscape

To reduce costs, you may want to combine the development of multiple add-ons in a single system landscape. This has the benefit of requiring fewer systems and saving hardware costs. There is less administration needed, which also reduces costs. The amount of quality assurance work needed, on the other hand, rises.

There are also other important limitations to be noted if you want to ensure that the add-ons developed in a single system landscape can function correctly.

Restrictions

- **Disjointness**

If you want to deliver add-ons independently of each other, you must develop them so that their objects are disjoint.

There is, however, no technical support for verifying the disjointness objects from different add-ons developed in the same system. There is a risk that add-ons reference each other, hence making them dependent. The consequence here is that your customers are not able to import these add-ons one at a time. The add-ons are also unable to run independently of each other.

Example

A data element from add-on 2 uses a domain from add-on 1.

The system does not check for disjointness during development, which means that some errors are not detected until the import test. Some other errors are not even found until runtime of the add-on (when functional tests are run or when the customer uses the add-on). This is the case, for example, if add-on 1 calls a function module or program from the (nonexistent) add-on 2.

Make use of the options in the package concept to structure your add-on development in such a way that objects are kept separate. For more information, see SAP Help Portal under [Package Builder and ABAP Package Concept](#).

- **Non-modifying development**

Add-ons **must not modify** other software components.

The reasons stated under *Disjointness* also apply to non-modifying development. If an add-on modifies another software component, the source state for further add-ons no longer matches. In these cases, there is the risk that add-ons are mutually dependent and cannot run without each other.

It is not possible to develop multiple modifying add-ons in the same system, since you need to test the technical installation of each add-on in a separate system. This is required to verify that the add-ons are independent of each other or that one add-on does not reference another. Moreover, external factors, such as separate delivery cycles, can prevent two modifying add-ons from being developed in one system at the same time. This is the case if add-on 1 requires an SAP support package to be imported, because this support package requires a CRT. Add-on 2, on the other hand, does not permit this import, since it is still in the development phase.

- **Importing supporting packages/upgrading systems**

Restrictions also apply when importing support packages in your system or when upgrading your system. Support package imports or upgrades are only possible centrally for each system. This creates a strong dependency between the release cycle and support package cycle and the add-ons.

➔ **Recommendation**

Due to these drawbacks, we advise against creating multiple add-ons in a single system. This applies in particular to add-ons that are independent of each other. If an add-on is based on another add-on, however, it may be a good idea to develop and maintain both in the same system. In this case, you can specify that one add-on is a prerequisite for the other add-on when you register it.

Notes for Developing Multiple Add-Ons in a System

If you want to develop multiple independent add-ons in a single system regardless, note the following points:

- Keep your work in the same prefix namespace. This helps you to ensure that an add-on does not reference the objects in a different add-on (but is only an indication and not fully reliable).
- Use the encapsulation methods available as part of the package concept. For more information, see SAP Help Portal under [Package Builder and ABAP Package Concept](#).
- When you create the delivery, use the [extended syntax check](#) for the delivery piece list in Transport Organizer (transaction SE01). Perform this check in the final assembly system and in the test system. This check verifies that the add-on is technically correct. Also use further check functions, such as those in ABAP Workbench (ABAP Test Cockpit or ABAP Unit Tests).
- For each add-on, set up its own test system and perform import tests and functional tests.

8.4 Client Layout and Transport Paths

General Information about Clients and Transport Paths

You can find fundamental information about clients and transport paths on SAP Help Portal in the documentation about [Change and Transport System](#) under:

- Change and Transport System - Overview
- Transport Management System (BC-CTS-TMS)

i Note

If not otherwise described, you always create new clients as a copy of client 000.

Recommended Client Layout and Transport Paths for the Development and Maintenance Landscape

Client Layout

Make the following settings (call transaction SM30 and display table T000):

Table 7:

System	Changes and Transports for Client-Dependent Objects	Changes to Cross-Client Objects
Development or Maintenance System		
Development client or correction client (client X00)	Changes without automatic recording	Changes permitted to repository and client-dependent customizing settings
Customizing client (client X05)	Automatic recording of changes	No changes to repository objects
Consolidation System or Consolidation System for Corrections		
Test client (client X00)	Changes without automatic recording; no transports allowed	No change to repository and client-independent customizing objects
Delivery client (client X05)	No changes allowed	No change to repository and client-independent customizing objects

Transport Paths

Make the following settings in the Transport Management System (TMS) for transporting your changes:

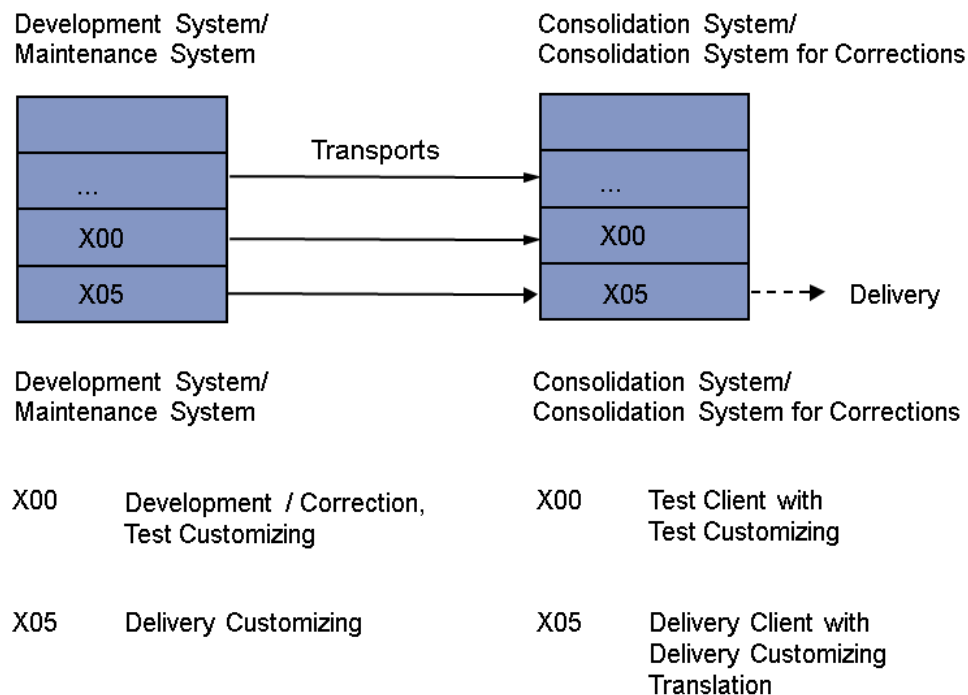
Source Client = Target Client

i Note

If you also want to test the delivery customizing in the test client, activate the enhanced transport control so that transport tasks from customizing client (X05) also reach test client (X00).

You can find information about enhanced transport control in SAP Help Portal under [Special Features of Enhanced Transport Control](#).

The following schematic graphic shows how you should create transport paths.



Read section [Setting Up Parameters LANGUAGE and LSM \[page 43\]](#) to find out about controlling the export of language-dependent data.

For more information, see [Defining Parameters for the Transport Control Program tp \[page 42\]](#).

8.5 Installing SAP Add-On Assembly Kit

Use

SAP delivers SAP Add-On Assembly Kit as a standalone add-on (AOFTOOLS). Use Add-On Installation Tool to import it into your system landscape.

Procedure

1. Before you install SAP Add-On Assembly Kit Version 5.0, read SAP Note [2179441](#).

2. Install SAP Add-On Assembly Kit in your development and consolidation system as described in SAP Note [2179441](#).
Pay particular attention to the preparation and follow-up actions specified in the note.
You require SAP Add-On Assembly Kit in the development system to be able to define the add-on software component and the add-on release. You require SAP Add-On Assembly Kit in the consolidation system too to create the delivery.

Result

You have installed SAP Add-On Assembly Kit. You can now configure the systems.

8.6 Making Settings for the Development and Consolidation System

You have to make the following settings:

[Entering the Namespace and Defining the Namespace Role \[page 39\]](#)

[Creating or Updating the Add-On Software Component and Add-On Release Name \[page 40\]](#)

[Setting the System Change Option \[page 41\]](#)

[Setting the Parameters for the Transport Control Program tp \[page 42\]](#)

[Setting the LANGUAGE and LSM Parameters \[page 43\]](#)

8.6.1 Entering a Namespace and Defining a Namespace Role

Use

You must work in defined namespaces when creating and maintaining add-ons. This requires you to enter the namespaces in a table and define the namespace role in the systems after reserving the namespace.

Prerequisites

Your user is assigned to the role `SAP_AAK_SDC_CHANGE`.

You have reserved one or more namespaces for the development objects (and a namespace for the add-on software component if required). Read the section [Reserving Namespaces \[page 23\]](#).

Procedure

To enter your reserved namespaces in the development system and consolidation system, proceed as follows:

1. Start Software Delivery Composer (transaction SSDC).
2. On the initial screen, choose ► *Environment* ► *Create/Update Namespace* ►.
3. On the screen *Change View "Repository Namespaces": Overview*, choose the *New Entries* function.
The view for editing the namespaces appears.
4. Enter the reserved namespaces and define the namespace role for each namespace.
This determines how the objects are handled in the namespace (for example, whether corrections are allowed or only repairs).
Set the namespace role as follows:
 - In the development and maintenance system, **P** (for the producer, who creates and edits objects)
Here you have to enter the development license for the installation number of the system.
You obtain the development license for the installation number of the system in SAP Support Portal. (You can find more information in the section [Reserving Namespaces \[page 23\]](#).)
 - In the consolidation systems (and hence in the test systems), **C** (for consumer, who only repairs the objects)
Here you must enter the valid repair license for the namespace.
You obtain the repair license for the namespace in SAP Support Portal. (You can find more information in the section [Reserving Namespaces \[page 23\]](#).)

Note

When creating or updating the software component and for creating deliveries in the consolidation system, you also require the value development license for the namespace you reserved for the add-on software component. (See also: [Creating Deliveries \[page 57\]](#)) To save the development license in the consolidation system, you can set the namespace role of the namespace of the add-on software component to **P** temporarily and enter the valid development license. Then change the namespace role back to **C**. The development license is now saved in the system. You can create or update the software component and create the delivery without entering the development license again.

8.6.2 Creating or Updating an Add-On Software Component and Add-On Release Name

Use

Before you can start development work in any new add-on release, you must create or update the add-on software component and the add-on release name in the **development system**. When you create packages for your development work, you must assign them to the software component.

The add-on software component is also a prerequisite for creating deliveries. This means that the add-on software component and the add-on release name must exist and be up-to-date in the **consolidation system** (also known as the final assembly system).

Prerequisites

- Your user is assigned to the role `SAP_AAK_SDC_CHANGE`.
- You have defined the add-on software component and the add-on release name. Also read the information under [Defining the Namespaces \[page 21\]](#).

Procedure

1. To create your add-on software component in the development system, start Software Delivery Composer (transaction SSDC) and choose ► [Environment](#) ► [Create/Update Add-On Software Component](#) ►.
2. In the [Software Component](#) field, enter the previously defined add-on software component.
3. If you have not saved the development license for the namespace, enter it now.
4. Enter the new add-on release and a description.
The system then enters the required data in the appropriate tables automatically.
A dialog box is displayed where the transport request is queried.
5. Create a new transport request or choose an existing request.
The request contains a transport entry for your add-on software component.
6. Release the transport request and import it into the consolidation system. This is necessary to make sure that the add-on software component, the add-on release name, and the associated description exist in the consolidation system.
7. In [translation \[page 50\]](#), verify that the description of the add-on software component is translated into all shipped languages. At the very least, the software component description must be available in English (even if the add-on does not exist in English), since English is SAP's standard language. When the delivery is created, Software Delivery Composer verifies whether the translated entries are in the delivery and whether the English entry exists.

Note

If SAP Add-On Assembly Kit is not installed in your development system, you can create the add-on software component and the add-on release name in the consolidation system. Then release the transport request containing this data and transport it back to the development system. In this way, you also register the add-on software component and add-on release name in the development system.

8.6.3 Setting the System Change Option

Use

The system change option dictates whether repository objects and cross-client customizing objects are modifiable or not.

For more information, see [Setting the System Change Option](#) in SAP Help Portal.

Prerequisites

Your user is assigned to the role `SAP_AAK_SDC_CHANGE`.







You have [entered your namespace or namespaces for the development objects \[page 39\]](#).

Procedure

Note

The system change option is dictated by both the namespace for the development objects **and** the software component.

Once you have released the namespace or namespaces for the development objects, you must configure the system change option in the development and maintenance system.

1. Call transaction `SE03`.
2. Choose  [Administration](#)  [System Change Option](#) .
3. To be able to modify objects from the namespace for the development objects, you must set both the global change option and the change option of the namespace to modifiable.
4. You must also set the add-on software component assign to the objects to [modifiable](#) to be able to modify the objects.
5. Also check the system change option of all other software components. To avoid modifying any software components accidentally, set all software components except your own add-on component to [not modifiable](#). If you are making modifications to another software component, you must also set this component to [modifiable](#) Restrict modifiability to as few areas as possible to avoid accidental modifications being made.
6. If you want to configure additional restrictions to the system change option in the maintenance system, you can define special roles and authorizations for your developers. In this way, for example, you can specify that developers are not allowed to modify any dictionary objects or UIs. For more information about roles and authorizations, see SAP Help Portal [User and Role Administration of Application Server ABAP](#) in SAP Help Portal or choose  [Help](#)  [Application Help](#)  when editing roles in transaction PFCG.

8.6.4 Configuring Parameters for the Transport Control Program tp

Use

Before you create and maintain add-ons, you must configure parameters for the transport control program `tp` in your systems.

For information about these parameters and how to configure them, see [Change and Transport System](#) under SAP Help Portal.

- [Transport Tools](#)

- [Changing Parameters for the Transport Control Program](#)

Procedure




When developing add-ons, you configure the parameters shown in the table for `tp` in Transport Management System (TMS) as follows:

Table 8:

Parameter	Value in Development System/Maintenance System	Value in Consolidation System/ Consolidation System for Corrections
ABAP/NTFMODE	b	b
NEW_SAPNAMES	true	true
T_IMPORT	no	no
K_IMPORT	no	yes
VERS_AT_IMP	yes	no

Note

The [Transport Tool](#) tab in TMS shows you only the automatically generated profile parameters for the transport control program `tp`. For a description about how to change these parameters, use the links to SAP Help Portal shown above.

To display all parameters, choose  [Goto](#)  [TP Parameters](#) .

8.6.4.1 Setting Up Parameters LANGUAGE and LSM

Use

Before you create and maintain add-ons, you must configure these two parameters in your systems.

You can control the export of language-dependent data using parameters `LANGUAGE` and `LSM` (`language_selection_mode`).

Note

You should only export from the development system in the master language, and from the consolidation system in all supported languages.

Procedure

Configure the following settings in Transport Management System the transport control program `tp`:

1. Select [Global](#) for parameter `LANGUAGE` and enter all relevant languages.
2. Set the parameter `LSM` as follows:

In the Development System

Enter the value **MASTER** for the `LSM` parameter.

The value **MASTER** means that language-dependent data is exported in the master language.




In the Consolidation System

Enter the value **ALL** for the `LSM` parameter.

The value **ALL** means that all languages that you entered for parameter `LANGUAGE` are exported.

With this value all table entries are exported according to parameter `LANGUAGE` without you having to add the missing languages to the language-dependent tables.

Note

You can also configure the export languages during the setup of the delivery in the Software Delivery Composer. To do this, choose  [Delivery Component](#)  [Select Export Languages](#) . The settings that you made in the Transport Management System are then overwritten.

8.7 Making Settings for the Maintenance Systems

The details included in the section [Making Settings for the Development and Consolidation System \[page 39\]](#) also apply for the settings in the systems in the [Maintenance Landscape \[page 32\]](#).

Note

You must make the same settings for the maintenance system as for the development system, and the same settings for the consolidation system for corrections as for the consolidation system for development.

8.8 Upgrading the Development and Maintenance Landscape

When upgrading a system landscape in which you are performing the add-on development or maintenance work, all objects that you have created yourself are retained. These are all those objects that have an object directory entry whose original system is not SAP.

If you have made modifications to SAP objects (objects with an object directory entry whose original system is SAP), these are overwritten by the SAP standard during upgrade. After the upgrade, however, you can reproduce your modifications during modification adjustment.

8.9 Test Systems

You must perform the following tests when developing and maintaining add-ons:

- Final assembly test (**mandatory**)
Test the imports of the finished packages here.
If any corrections are required, you must make them in the development system or maintenance system and import them into the consolidation system. You can then create an improved package. You then test the import of the new package again.

➔ Recommendation

We recommend that you make backups of the various system states. You can then import a specific state into the final assembly system for testing.

- Acceptance test (**optional**, but recommend for large scale development projects)
After development close, corrections are passed continuously to a test system that contains test data. The aim of this is to test the add-on functions and customizing changes continuously in the final stages of development.

⚠ Caution

When add-on functions or customizing changes are tested in the consolidation system, there is the risk of passing cross-client test data to the delivery.

➔ Recommendation

We recommend that you use an acceptance test system located after the consolidation/final assembly system in the system landscape and which, unlike this system, contains extra cross-client test data. You can use this test system, for example, as a consumer system after the consolidation system. Make your settings (such as the settings in the transport profile) in the same way as in the consolidation system.

Ideally, two systems should be provided for these tests. However, if time constraints and your development process dictate otherwise, you can also perform these tests in a single system, but at different times.

It is essential, however, that you have a (temporary) test system for final assembly tests, which you update to the version where you want to test a delivery using system copies created in advance.

9 Add-On Development

You can find basic information about ABAP development on SAP Help Portal under [Application Development on AS ABAP](#).

Develop your objects in the development system. Develop all cross-client objects in the development client and client-specific objects in the customizing client.

When you create new objects, verify that the correct original language is set. We recommend that you always log on in the development language, since the default original language of an object is the same as the logon language. If you log on in a different language, this can cause problems in translation or in customer deliveries with multiple languages.

Once a development phase is finished, transport the objects in question to the consolidation system.

Translation takes place in the delivery client of the consolidation system.

As specified by your delivery strategy, import the current support packages into your development system (the development system must have the SAP support package state dictated by the customer system).

9.1 Creating and Assigning Packages (Transaction SE80)

To develop your add-ons, you have to create packages. Assign each package of the add-on to the relevant software component and to a transport layer. Prerequisite for assigning packages is that you have set up a software component in the development and consolidation systems for each add-on release, as well as a transport layer for the consolidation transports.

9.2 Rules for Add-On Development

Follow these rules when developing your add-on:

- **Platform-neutrality**



All add-on-specific developments must conform to the SAP platform strategy. ABAP developments must support all application servers on the platforms UNIX, Windows, and IBM eServer iSeries. Exception are possible only when unavoidable for technical reasons.

Note the following technical points:

- The use of `call system` in ABAP programs points to platform-specific applications. An alternative here is to use platform-specific commands defined in the transaction `SM69`.
- An application runs either on the ASCII code page or EBCDIC code page, as specified by the platform. This means that logical comparisons like `f1 BETWEEN f2 AND f3` can produce different results on different platforms.
- Note the code page used when the file system of the application server is accessed.

- If you want to call external programs using CPIC communication, you must implement a code page conversion. RFC communication incorporates this conversion automatically.
- Develop the add-on as a **self-contained functional unit** with well-defined interfaces to the SAP software.
- **Do not modify the SAP software.**
Use [enhancement techniques \[page 48\]](#) instead. If you want to make modifications regardless of this, follow the [rules for developing modifying add-ons \[page 119\]](#).
- **Develop the add-on in such a way that it can also be uninstalled.**
Add-ons no longer in maintenance or no longer needed by customers should be removed from the system. The add-on must be developed to take this into account. For more information about this, see [Add-On Uninstallation \[page 81\]](#).
- **Develop only one add-on in each system landscape.**
If you want to develop more than one add-on in a system regardless of this, read the notes in [Developing Multiple Add-Ons in a System Landscape \[page 35\]](#).
- **Client-specific objects:** Use **BC Sets** and **IMG enhancements**.
For more information about this, see SAP Help Portal under ► help.sap.com ►

Table 9:

BC Sets	Business Configuration Sets (BC-CUS) 
IMG enhancements	IMG Enhancement 

- **Tables**

Content of SAP Tables

If you want to deliver entries in SAP tables, use BC Sets.


If you deliver table entries directly, without using BC Sets, this counts as a modification that cannot be adjusted. These entries can be overwritten at any time and without warning in your system and in the customer system by SAP support packages or upgrades.

You cannot use AAK to deliver entries in ranges reserved for customers in SAP tables, since these entries could overwrite the content in the customer system.

New Tables

If you want to create a new table, make sure that you assign it the correct delivery class. The delivery class specifies the transport behavior of tables. Only tables in the delivery classes E, G, and C support BC Sets. Note also that further restrictions apply when using BC Sets (such as restrictions on the length of the table name).

For more information about delivery classes of tables, choose the information button in ABAP Dictionary (transaction SE11) in your SAP system or under [Delivery Class](#) in SAP Help Portal.

- Also note the instructions under [Object List Checks \[page 61\]](#) when developing objects.
- If your add-on supports multiple SAP releases, also read [Rules for Add-On Development on Multiple SAP Releases \[page 47\]](#).
- Your development takes place in a prefix namespace, which means you must also read the information about restrictions on development in namespaces in SAP Note [104010](#) .

9.2.1 Rules for Developing Add-Ons in Multiple SAP Releases

It is important to observe the following rules if you are developing your add-on in multiple releases.

- **Originality/New Objects**

If you need your add-on objects in multiple SAP releases, you should always create the new objects in the lowest supported SAP release. This gives you the following benefits:

- The upward compatibility of the objects is generally guaranteed. This means, for example, that the ABAP syntax normally remains valid in the higher release, and the objects types are normally known in the higher release too.
- Transporting copies from a lower SAP release to a higher one is usually possible from a technical point of view.
- Having the objects in the same system as the original system helps you to keep an overview of all add-on objects.

As far as possible, you should therefore always create the new objects in the development system in the first development landscape. If you need the objects in another development system, you can transport copies to the development system in the new landscape. Note that you might need to make adjustments to the objects in the higher release.

If you do not need an object in the first development system, you can create it in the next higher development system in the new development landscape.

Make sure that objects always only have one original system and that they only exist as copies of this original in any other systems.

- **Transporting Copies of Add-On Objects**

When transporting copies between development systems in various releases, you should always transport them from a lower SAP release to a higher one. The reasons for this are explained under Originality/New Objects .

- **Modifications**

If you need modifications in the higher SAP release, you have to perform these manually. You should not transport copies with modified objects, as you cannot assume that the status of the modified objects in the higher SAP release corresponds to that in the lower release.

- **Non-Release Specific Development**

To keep down the costs and effort required for developing your add-on in multiple release tracks, your development should be non-release specific wherever possible . For details, see [Determining the Delivery Strategy \[page 19\]](#) .

9.3 Additional Information About Enhancements and Modifications

Enhancement techniques are ways of expanding the SAP software without making modifications. If you want to make modifications, we recommend using the Modification Assistant.

More information about these topics is available in the **SAP training course BC425** and in SAP Help Portal under [Changing the SAP Standard \(BC\)](#).

If your development work is based on SAP NetWeaver 7.0 and higher, you can use Enhancement Framework. More information can be found in SAP Help Portal under [Enhancement Framework](#).

i Note

The information that follows focuses on individual objects, but does not claim to be complete.

Enhancement Technique

Enhancement Spots

Use enhancement spots as your preferred enhancement technique. These are spots in the source code defined by SAP where you can insert code without modifying the original object. You can choose to expand on the original logical or add a ready-made implementation.

For more information about enhancement spots, see SAP Help Portal under [Enhancement Spots](#)

Notes for Essential Modifications to Standard SAP Objects

Dictionary Objects

When developing add-ons and modifying dictionary objects, particular care should be taken when making enhancements to tables.

- **Enhancements to tables**

Never enhance standard SAP tables by adding new add-on-specific fields, since these modifications can be overwritten by updates to the table object (in support packages or upgrades), by SAP software components, or by other add-on components in the add-on system.

Use the append structure as an enhancement method for standard SAP tables. This makes it possible to enhance tables without making modifications and the tables are not overwritten when the table object is transported again from the standard SAP system. This type of enhancement is not possible for pooled tables or cluster tables.

For more information about append structures in tables, see SAP Help Portal under [Append Structures](#).

Programs

Use Modification Assistant to make changes to programs. Activate Modification Assistant in your development system.

For more information about Modification Assistant, see SAP Help Portal under [The Modification Assistant](#)

➔ Recommendation

We recommend that you **do not** make any modifications.

- **Function Modules**

It is usually better to call a function module that encapsulates the enhancements in the SAP program instead of including a statement block directly in the program.

If a new function module like this is developed in an add-on development system, it must be created within a separate add-on function group. This prevents the new function module from being overwritten by any updates. Potentially only the call of the new function module needs to be inserted in the source code.

If you want to insert code in an existing standard SAP function group to use its global memory, use a form routine instead of a function module. This avoids any inconsistencies when assigning include numbers to the function modules. It is not possible to create new function modules in SAP function groups.

- **Form routines**

If possible, avoid using any `PERFORM` calls and use function modules instead. The only exception is when creating form routines instead of function modules in a standard SAP function group, to enable access to the global memory of the function group.

- **Variables**

If you need to create a new variable within an SAP program, you must define this variable locally.

- **Messages**

You must use separate message IDs exclusively for the add-on when creating new messages within an SAP program.

You cannot create messages in existing SAP namespaces.

9.4 Documentation and Translation

Documentation

Software development also involves writing documentation. Documentation is required to make it easier to user the add-on. You can write the documentation directly in the development system (this is known as online documentation).

For information about writing documentation, see SAP Help Portal under [Services for Information Developers and Translators](#).

Translation

If you want to deliver your add-on in more than one language, you must translate all language-dependent objects in the add-on. Alongside the documentation, this includes other objects (such as UI texts).

Before you can use the translation environment, you must first prepare your system for translation. Translation usually takes place in the consolidation system for development and corrections. If you have a large system landscape, you can also set up a separate translation system.

For information about translation, see SAP Help Portal under [Services for Information Developers and Translators](#).

Note the following points when organizing translation:

- For each new release, verify that the description of the add-on software component is translated into all shipped languages. To this, choose the following function in the translation editor (transaction SE63):
 1. Choose ► [Translation](#) ► [ABAP Objects](#) ► [Short Texts](#) ►.
 2. In the object type selection under [OO Meta Objects](#), select the type [TABL Tables \(Meta\)](#).
 3. In the object name field, enter [CVERS_REF](#). This is the name of the table in which the software component description is saved.
 4. Choose [Edit](#).
 5. In the [Software Component](#) field, enter the name of your add-on software component (or select it using value help) and choose [Continue](#).
 6. Translate the description of the software component.
- Before you deliver the add-on, you must make sure that all language-dependent objects in the add-on are translated.

-
- To export the translate languages correctly, [configure the parameters LANGUAGE and LSM \[page 43\]](#). This includes the languages in the add-on delivery immediately.
 - If you require an additional language for an add-on after you have delivered it, proceed as described in [Delivering Languages Translated Retroactively Translated \[page 71\]](#).

10 Delivering the Add-On Software

Once the development of a new add-on release or corrections for a new add-on support package level have been finished or the migration to a new SAP release has been completed, you must create the required delivery packages.

The following package types are available here:

- Initial delivery of the add-on: *Add-on installation package*
- New support package level of the add-on: *Add-on support package*
- Adjustment with SAP support packages: *Conflict resolution transport*
- Delta upgrade of the add-on software: *Add-on upgrade package*
- Exchange upgrade of the add-on software (includes it in the SAP system upgrade): Either an *add-on exchange package* (package type AOX) or an *add-on upgrade package*, depending on the underlying SAP release

Deliveries are created in a series of steps:

1. Compose the delivery

You define the type of the delivery in Software Delivery Composer. You then create the associated delivery request for the delivery and fill them with content. Checks in Software Delivery Composer consolidate the content of the delivery requests.

The following delivery types and delivery requests are available:

- Delivery type *add-on installation/upgrade* :
The following delivery requests are available here:
 - *Change piece list*
 - *Component piece list*
 - *Exchange component piece list*
- Delivery type *support package*:
The delivery request *support package* is available here.
- Delivery type *conflict resolution transport*:
The delivery request *conflict resolution transport* is available here.

2. Create the delivery package

You use Software Delivery Assembler to do this. Software Delivery Assembler converts each delivery request into an importable package. This table shows you the package types into which the delivery requests are converted:

Table 10:

Delivery request	Package type
Change piece list	Add-on upgrade package for add-on delta upgrade (AOU)
Component piece list	Add-on installation package (AOI)
Exchange component piece list	Dependent on the SAP release: Add-on exchange package (AOX, from SAP Net-Weaver 7.0) or add-on upgrade package (AOU, up to SAP Web AS 6.40) for add-on exchange upgrade

Delivery request	Package type
Support package	Dependent on the SAP release: Add-on support package of type CSP (from SAP Web AS 6.40) or AOP (up to SAP Web AS 6.20)
Conflict resolution transport	Conflict resolution transport (CRT)

3. Provide the delivery package

You can provide your customers with the importable package as follows:

- As a compressed file in the Internet
- On a CD with a predefined directory layout (see [CD for the Add-On Delivery \[page 130\]](#))

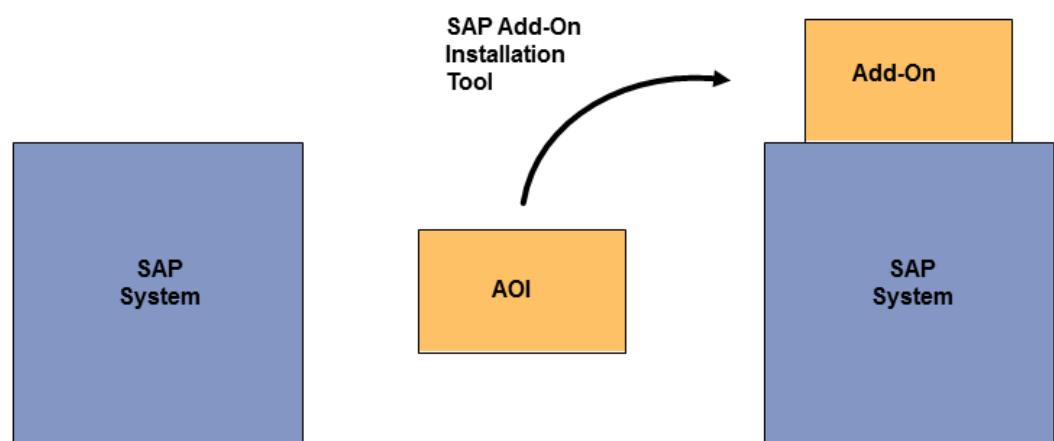
10.1 Package Types

The following package types can be used for deliveries:

- **Add-on installation package (AOI)**

Add-on installation packages are used for the initial delivery of an add-on. SAP Add-On Installation Tool is used to import add-on installation packages.

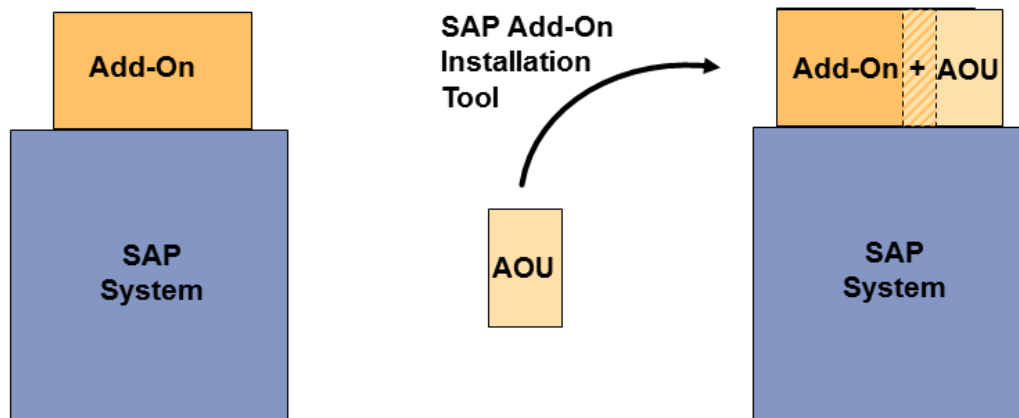
Add-On-Installation Package



- **Add-on upgrade package (AOU) for add-on delta upgrade**

An add-on upgrade package for the add-on delta upgrade is used to update the add-on when the SAP release does not change. SAP Add-On Installation Tool is used to import add-on upgrade packages.

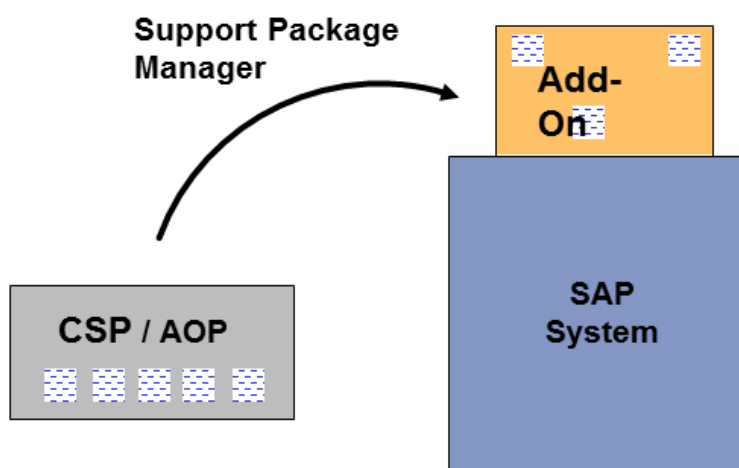
Add-On Upgrade Package for Add-On Delta Upgrade



- **Component support package (CSP) or add-on support package (AOP)**

An add-on support package is used to change the add-on support package level. Depending on the underlying SAP release, the package type is either CSP (for component support packages from SAP Web AS 6.40) or AOP (for add-on support packages up to and including SAP Web AS 6.20). The term add-on support package is used for both package types in this documentation. Support Package Manager is used to import add-on support packages.

Add-On Support Package

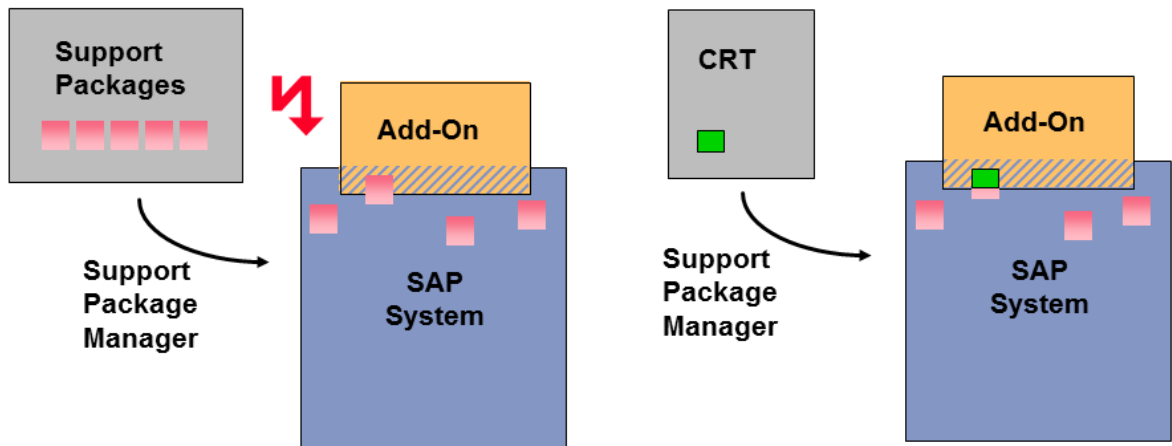


- **Conflict resolution transport (CRT)**

A conflict resolution transport is used to recover modifications in a modifying add-on if they were overwritten by an SAP support package. A conflict resolution transport can also contain corrections to the add-on itself. Support Package Manager is used to import conflict resolution transports.

For more information about modifications, see [Modifications and Their Consequences \[page 115\]](#).

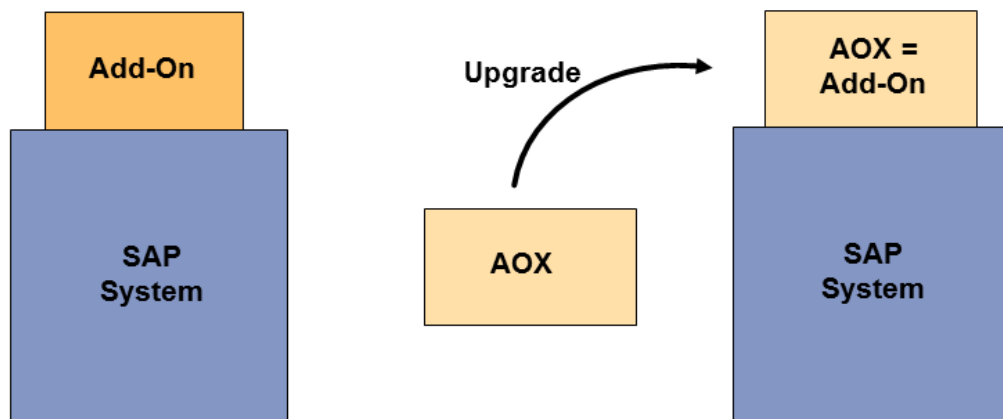
Conflict Resolution Transport



- **Add-on exchange package (AOX)**

An add-on exchange package for the add-on exchange upgrade is used to update the add-on at the same time as an SAP system upgrade. During an SAP system upgrade, add-on exchange packages are included in the upgrade by the SAP upgrade tool. You can create add-on exchange packages for add-ons based on SAP NetWeaver 7.0 and higher.

Add-On Exchange Package



SAP System in Source Release

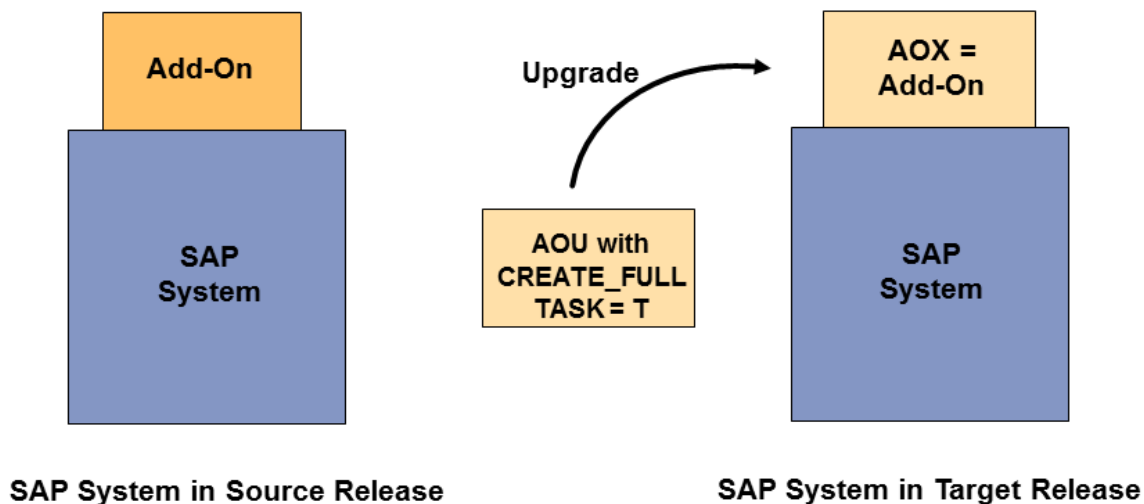
SAP System in Target Release

- **Add-on upgrade package (AOU) for add-on exchange upgrade**

An add-on upgrade package for the add-on exchange upgrade is used to update the add-on at the same time as an SAP system upgrade. Add-on upgrade packages must contain the extended attribute `CREATE_FULLTASK = T` and are include in the upgrade by the SAP upgrade tool.

You can create add-on upgrade packages for the add-on exchange upgrade for add-ons based on SAP Web AS 6.20 and SAP Web AS 6.40.

Add-On Upgrade Package for Exchange Upgrade



10.2 Final Assembly System

As described in [Creating the System Landscape and the Systems \[page 29\]](#), you can use the consolidation system as the final assembly system. This means that the details included in [Making Settings for the Development and Consolidation System \[page 39\]](#) also apply to the settings for the final assembly system.

Note

Note that the settings for the consolidation system also apply to the final assembly system in the maintenance track.

See also:

[Creating the System Landscape and the Systems \[page 29\]](#)

[Making Settings for the Development and Consolidation System \[page 39\]](#)

[Making Settings for the Maintenance Systems \[page 44\]](#)

10.3 Creating Deliveries

Once you have finished developing, maintaining, or updating your objects, you can create the delivery using the SAP Add-On Assembly Kit tools.

Context

The way in which the delivery is created is specified by the type of the delivery and hence the package type. The procedure that follows describes the general steps performed when creating a delivery. For specific information about the delivery types or package types, see the relevant sections (*Creating an Add-On Installation Package* and so on) or in the online documentation of the tool in question. This is accessed by choosing the help function in the pushbutton toolbar.

Procedure

- **Composing a Delivery Using Software Delivery Composer (SDC)**

Before you use SDC, import the highest available support package of SAP Add-On Assembly Kit.

i Note

For this, you need the role `SAP_AAK_SDC_CHANGE`.

- a. When you begin, define the type of the delivery (add-on installation/upgrade, add-on support package, or conflict resolution transport) and other administration data in SDC. Here, the namespace prefix of the delivery name is used automatically as the add-on software component.

To identify yourself as the owner of the namespace of the delivery, you may need to enter the associated development license. This guarantees that deliveries can only be created by the owner of the namespace.

For more information, see [Entering a Namespace and Defining a Namespace Role \[page 39\]](#).

- b. Create the list of all modified objects in the delivery (the change list) as the first delivery request. Fill the change list using various selection criteria for change requests (or objects). Flag the change requests in question first and then include them in the list.
- c. Verify in which languages the delivery request is exported or select the languages. To do this, choose **► Delivery Component ► Select Export Languages ►**.
- d. Use SDC to perform the [object list checks \[page 61\]](#). Make any corrections needed to the change list.
- e. Once you have specified the content of the change list, release it in SDC.
- f. Depending on the type of the delivery, you need to create further delivery requests (such as component piece lists and exchange component piece lists). Perform the object list checks for these too and release the delivery requests once their content is consistent.
- g. Release the associated delivery component.


- **Assembling a Delivery Package with Software Delivery Assembler (SDA)**

Note

For this, you need the role `SAP_AAK_SDA_CHANGE`.

- a. You can register the delivery request in the following ways:
 - Start the registration from SDC
In SDC, select a delivery and then the delivery request that you want to register. Choose **► Delivery Request ► Register (by SDA, Locally) ►**. SDA takes you to the tab page of the package type in question. The system takes the package attributes and import conditions from SDC.
 - Direct import with SDA
Start SDA and select the tab page of the package type associated with the delivery type. Enter the name of the delivery request in the *Package* field and the value **NONE** in the *RFC Destination* field. Enter values for *Add-On* and *Add-On Release* (in CSPs, *Component* and *Component Release*) and choose *Import*.
 - Direct import with template with SDA
Start SDA and select the tab page of the package type associated with the delivery type. Enter the name of the delivery request in the *Package* field and the value **NONE** in the *RFC Destination* field. Choose **► Select ► Registration Options ►**. In the next dialog field, choose *Package Template* and enter the name of the package that you want to use as a template for attributes, extended attributes, and import conditions. Choose *Continue*.
- b. Check the attributes and import conditions generated by the system. Add new attributes and conditions, change them, or delete any entries no longer relevant.

Note the following points here:

- In SDA, specify the conditions to be checked by the import tool in the target system. These include the release and support package levels of software components or prerequisite software components. Also include all dependencies.
- Respect any requirements that are specific to your delivery type. For more information, see the sections about the relevant package type ([Creating an Add-On Installation Package \[page 68\]](#), [Creating an Add-On Support Package \[page 76\]](#), [Creating a Conflict Resolution Transport \[page 120\]](#), [Creating an Add-On Upgrade Package \[page 77\]](#), [Add-On Behavior in SAP System Upgrades \[page 78\]](#)) or the online documentation in SDA.
- If you add the extended attribute `SEE_PNOTE`, the import tool requires a password when the package is imported. This password is usually provided in an SAP Note. Customers have access to this note. The SAP Note [567695](#)  is provided as a fallback, since it is not usually possible for you to create notes. This note instructs customers to contact their add-on producer to obtain the password. Notify your customers of this password, for example, in the installation guide of your add-on. You can generate the password in SDA by choosing **► Extras ► Password Generation ►**.

Recommendation

We recommend that you always use this attribute. This enables you to give your customers additional information about the package.

- If you require a specific version of Support Package Manager/SAP Add-On Installation Tool to import your package, you can use the extended attribute `NEED_SPAM_LEVEL` to define that at least the version in question has to exist.

➔ Recommendation

We recommend that you always choose the current version of the SPAM/SAINT update. You can download it from the Software Download Center in SAP Support Portal.

- c. Register the delivery request.

This provides you with an importable package. The package has the status [Locked](#). It is located in one of the following directories:

- In the subdirectory `...EPS/out` of the transport directory (for example `/usr/sap/trans/EPS/out`). The transport directory itself is defined in the profile parameter `DIR_EPS_ROOT`.
- In the subdirectory `.../out` of the path defined by the profile parameter `DIR_EPS_ROOT` (for example `/usr/sap/trans/EPS`). The value for the root path in `DIR_EPS_ROOT` (for example `/usr/sap/trans`) does not need to be the same as the transport directory.

You require administrator rights to access these directories.

- d. Optional: Create an SAR archive.

If you do not have administrator rights, you can download the importable packages as an SAR file. To do this, switch to the [Administration](#) tab page. Enter the name of the package and define the target folder on the local computer if necessary. Then choose [SAR Download](#).

• Testing the Delivery

- a. Make the package available for a test system.

To do this, go to the transport directory `EPS/out` in the final assembly system. Copy the delivery package in the test system into the transport directory `EPS/in`.

- b. If you want to test imports of the package using SAP Add-On Installation Tool or Support Package Manager, upload the package into the test system.

To do this, go to Support Package Manager in the test system and choose ► [Support Package](#) ► [Load Packages](#) ► [From Application Server](#) ►.

The package is available for imports in the test system.

i Note

This step is not required if you want to test how the package is included in SAP system upgrades, since the SAP upgrade tool loads the package from the transport directory itself.

- c. Then import the package in your test system.

Use the appropriate import tool to do this.

- d. Test the functions in the package.

• Removing Errors in the Delivery

You can remove errors in the delivery in the following ways:

- If tests detect errors in the attributes or import conditions for the package or if attributes or import conditions are missing, you can correct them in Software Delivery Assembler and register the package again. Proceed as follows:
 1. In Software Delivery Assembler, go to the tab for your package type and enter the name of the package.
 2. Choose [Change Attributes](#) in the registration options.

3. Correct the errors in the attributes and import conditions.
 4. Register the package again.
 5. Test the delivery package.
- If tests detect errors in the content of the delivery request or if content is missing, you can undo the release of the delivery component in Software Delivery Composer. This also undoes the release of the delivery requests and you can make any corrections to the content of the delivery request in question. Proceed as follows:
 1. Reset the delivery component in Software Delivery Composer.
 2. Correct and check the delivery request.
 3. Release the delivery request again.
 4. Release the delivery component again.
 5. Register the delivery request again using Software Delivery Assembler.

To do this, go to the tab for your package type in Software Delivery Assembler, enter the name of the package, and choose [Exchange Data File](#).
 6. Test the delivery.

• Post-Delivery of Attributes

After the delivery of packages (in add-on installations, upgrades, or support packages), it is often necessary to correct import attributes or add new attributes. For example, you may need to add new import prerequisites to validate additional system states as a basis for imports.

The package type *attribute change package* (ACP) makes it possible to deliver these modified attributes. When attributes in a released package are modified, the system creates an ACP automatically for each software component version. If the attributes of multiple released packages in the same software component version are modified, the system creates one ACP version for each package.

The name of an ACP has a maximum of 20 characters and is created from the name of the software component version: Characters 1 to 10 contain the name of the software component (filled with equals signs up to the 10th character if necessary) and the version of the software component is inserted from the 11th character.

Example

The ACP name for packages in the software component version SAP_APPL 600 is `SAP_APPL==600`.

To create an ACP for a released package, proceed as follows:

- a. Go to the tab of the package type for which you want to post-deliver attributes.
- b. Enter the package name and choose [Select Registration Option](#).
- c. Choose [Post-Delivery](#) and confirm by choosing [Continue](#).
- d. Make your changes in the attribute editor:
 - If you want to modify the extended attributes, choose the [Extended Attributes](#) tab.
 - If you want to modify the import conditions, choose the [Import Conditions](#) tab.
- e. Confirm your changes by choosing [Continue](#).
- f. Choose [Register](#).
- g. Confirm that they are correct by choosing [Yes](#).
- h. Confirm the EPS file by choosing [Continue](#).

An ACP can be delivered as a standalone package or together with another package (with a package type other than ACP). This is known as a container package.

- If you want to deliver the ACP as a standalone package, go to SDA and choose the [Administration](#) tab, enter the name of the ACP and release the package. When queried for a carrier package, choose [Without carrier package](#). The benefit of this solution is that changes can be made available immediately.
- If you want to deliver the ACP with a carrier package, you can proceed as described above. In this case, however, you select a suitable package when queried for a carrier package. Alternatively, you can decide not to perform a special release of the ACP. The system releases the ACP automatically when the next carrier package is released. Both packages (the ACP and the carrier package) must be made available for the import.

- **Finishing the Creation of the Delivery**

Once you have completed the delivery tests successfully, you can finish the creation of the delivery.

- a. Confirm the delivery in SDC.

You can no longer make any changes to the delivery in SDC.

- b. Then release the package in SDA.

This documents that the package has a defined state. The package and its associated attributes and import conditions can no longer be modified.

10.3.1 Object List Checks

When you create the delivery, check the object list of the associated delivery request to avoid any problems later when installing or maintaining the add-on.

You cannot add all objects in these requests to the delivery automatically.

Software Delivery Composer supports you when performing the object list checks.

Note


For more information about the individual object list checks, see the results display of the checks in Software Delivery Composer. Choose the question mark icon next to a check.

SAP delivers its own object list checks, but you can also define your own checks. For more information, see [Adding Custom Object List Checks \[page 64\]](#).


Checks in Software Delivery Composer

- **Forbidden objects**

Some objects are fundamentally invalid and cannot be delivered, for example because they are customer objects).

Customer objects are reserved for customer themselves. None of these objects can be delivered, since the customer uses live entries that could be overwritten by a delivery. For more information, see SAP Note [16466](#) .

Entries of the type R3TR VERS (definition of software components), for example, cannot be delivered because these entries are made by the import tools.

For a list of the forbidden object types and objects, see SAP Note [870407](#) .

- **Extended DDIC check**

The delivery can contain only consistent dictionary objects. Any inconsistent dictionary objects can have negative consequences, such as generation errors or even loss of data.

- **Existence of the delivered objects and table entries**

The associated object directory entries must exist for objects with an object directory entry. The objects must also be in a known and delivery-enabled package (also known as development class).

Another check verifies whether any deleted objects are in the delivery. It is not usually necessary to deliver deleted objects for an add-on installation.

- **Table entries: Generic transport**

Table entries (TABU) and logical objects cannot be delivered with simple generic keys (*). This would delete the full key range covered by the generic key in the customer system and replace it with the exported entries. The check detects the following table entries, which transport all table ranges:

- *: Full table content
- <language ID>*: Language-specific table content
- <client>*: Client-specific table content
- <client><language ID>*: Client-dependent and language-dependent table content

- **Required objects**

In the case of deliveries with type *Installation/Upgrade*, a check verifies that the delivery contains all definitions of packages (development classes) and namespace definitions for which objects exist in the delivery request.

For all delivery types, a check verifies whether the delivery contains namespace definitions for which there are no objects in the delivery request. For all namespaces in the delivery request, a check verifies whether a valid repair license exists.

Note

Your add-on customers can make changes to objects in your namespace only if a valid repair license is entered in the system. If you entered your repair license in your final assembly system, it is delivered with the add-on. If not, you must notify your add-on customers about the license, so that they can enter it in their system.

- **Table entries: Delivery-relevance/delivery behavior**

You cannot delivery any entries in tables with class A. Also check whether the other delivered table entries and logical objects demonstrate the correct import behavior (as dictated by the delivery class of the tables). In the associated check, Software Delivery Composer lists all table entries and logical objects with their import behavior.

The import behavior for installations and upgrades is different here for client-specific tables and cross-client tables.

- **(Invalid) modifications and deliveries**

Recommendation

We recommend that you only create **non-modifying add-ons**.

Avoid making modifications to objects from a software component that is not the modifying software component. Software Delivery Composer displays objects from software components that are not the modifying software component.

Note

Any modified dictionary objects that are used in transparent tables can cause updates to be made. Check for this behavior, since it can cause longer runtimes in upgrades and installations.

Caution

Do not modify non-versionable objects in other software components.

You are also not allowed to modify more than one software component.

- **Table entries in the customer range (protected in accordance with TRES)**
Table entries and logical entries cannot be in ranges reserved for customers (in accordance with table TRES).
- **Add-on uninstallation: Non-deletable objects**
It is not possible to delete every category of data from the system in add-on uninstallations. This check searches the delivery request for any objects and table entries that cannot be deleted. If these objects are delivered, the add-on cannot be uninstalled.
- **Generic checks and automatic corrections**
The content of the delivery request is checked for obsolete objects. Any changes needed to the current object types (for example, replacements by logical transport objects) are made automatically. No further manual modifications are necessary. These checks are always performed.




Additional Software Delivery Composer Check for Deliveries of Type Installation/Upgrade

- **Description of the software component (LANG VERS) required**
The delivery request must contain the add-on software component in English and in all delivered languages.

Additional Software Delivery Composer Check for Deliveries of the Type Support Package and Conflict Resolution Transport

- **New objects (entries) in support package/CRT**
Avoid delivering new objects in add-on support packages and conflict resolution transports (CRTs). Software Delivery Composer checks whether objects like this are delivered.

Note

Software Delivery Composer can perform this check only if the component piece list for the add-on release is known in your SAP system. If you created this piece list in a different system, you can register it with Software Delivery Composer in the current system. To do this, start Software Delivery Composer and choose  [Delivery for Delivery Request](#)  [Register](#) .

The following applies to CRTs: You can create a CRT for one main component or application component only, which means that the CRT can contain only objects from its own add-on software component or from the

assigned main component or application component. Software Delivery Composer checks whether the delivery request contains any objects from further software components.

Manual Check

The objects must be consistent. This means that, for example, all subobjects (such as program code) must exist as active objects. If not, errors occur when the package is imported (at the latest).

You can run this check before releasing the delivery request in Transport Organizer:

1. Call Transport Organizer (transaction `SE01`).
2. On the *Display* tab, enter the name of the delivery request and choose *Display*.
3. Choose **Request/Task** > *Complete Check* > *Objects (Syntax Check)* .

Perform this check both in the final assembly system and in the test system.

Additional Check when Releasing Delivery Requests for Deliveries of Type Installation/Upgrade

The delivery requests for deliveries of type *Installation/Upgrade* can contain only full objects from your add-on (of the modifying software component). This means that any subobjects (LIMUs) specific to the add-on must be replaced by the associated full objects.

When the change piece list, component piece list, and exchange component piece list are released, Software Delivery Composer replaces the subobjects specific to the add-on with the full objects automatically.

Caution

This does not apply to modified standard SAP objects.

10.3.1.1 Adding Custom Object List Checks

In Software Delivery Assembler, you can add your own object list checks that apply when creating deliveries.

If you need further object list checks for delivery requests, you can specify them as separate classes with a defined interface, known as check classes. Once these checks are activated, they are applied in every object list check in the delivery system. The displayed list of critical objects and table entries is updated accordingly.

Note

Alternatively, you can continue to add your custom object list checks using function modules in Software Delivery Assembler. The procedure is described in SAP Note [215178](#) .

Activating Additional Checks

To activate your custom checks, go to Software Delivery Composer and choose **Utilities > Add Object List Checks** for the delivery in question. Confirm the dialog box. On the next screen, choose **New Entries**.

Specify the following information for each check:

- **Name of Check Module**
Enter the name of your check here. This name can also be the name of the check class.
- **Short Text for Check**
Enter a description of your check in this field. This text accompanies this object list check on all UIs.

Caution

This text is language-specific. If you want to provide this text in a language other than the current logon language, first save your input. Then choose **Goto > Translation** and select the language in question. Enter the translation of the short text and save your input.

- **Class/Interface**
Specify the name of the check class (its regular ABAP OO class name) here.
- **Name of Documentation Module**
Enter the name of the long text.
You can create a long text (with the type dialog text, see transaction SE61) for each check. Use this long text to explain how the check works and how to respond to any errors.
If you want to provide the long texts on other languages, you can translate it in the translation editor (transaction SE63). For more information here, see [Services for Information Developers and Translators](#).

Notes:

- The order in which the object list checks are performed is not defined and can vary from system to system.
- Any changes to the request content (using database access to the table of the object list or the key entries) in the check classes for the object class itself are not allowed. Instead, the action in question is displayed in the results list of the object list check and the corresponding entry can be corrected from here if needed.
- The list of object list checks delivered in Software Delivery Composer can be expanded in later releases.
- Only use names from the partner/customer namespace as names of check classes. This avoids conflicts with the delivered object list checks.
- The checks are performed on varying object sets in a request.
 - Objects
 - Table keys
 - String-like table keys

Description of the Check Class and the Interfaces

The check class is a regular ABAP OO class. A class can be used as a check class in the context of object list checks when the interface `IF_EM_OLC` is implemented.

IF_EM_OLC

This interface provides a method, `CHECK`, and various events used in the check to indicate errors.

The check itself is performed in the implementation of the interface method `IF_LM_OLC~CHECK`. Here, access to the data of the request in question is provided by the importing parameter of type `IF_EM_REQUEST`.

IF_EM_REQUEST

The request has various methods that provide the content of the transport request in question:

- `GET_HEADER`
Gets the header entry of the request (structure like in table E070)
- `GET_OBJECTS`
Gets the list of objects in the request (table with the structure like in table E071; values of the fields `PGMID`, `OBJECT`, and `OBJ_NAME` (and `LANG OBJFUNC` if required)).
- `GET_KEYS`
Gets the list of table keys for objects in the request (for example, for `TABU` objects) (structure E071K); values of the fields `PGMID`, `OBJECT`, `OBJNAME`, `MASTERTYPE`, `MASTERNAME`, and `TABKEY` (and `LANG` if required)).
- `GET_STRINGKEYS`
Gets the list of string-like table keys for objects in the request (structure E071K_STR); values of the fields `PGMID`, `OBJECT`, `OBJNAME`, `MASTERTYPE`, `MASTERNAME`, `KEY_LENS`, and `TABKEY` (and `LANG` if required)).

Implementation of the method `IF_EM_OLC~CHECK`

You make the checks in the check method. The objects in question are provided using the method of the interface `IF_EM_REQUEST` (see above). Objects can have the following levels of severity:

- Severity:
 - **E** for error
 - **W** for warning
 - Other values for information; recommended: **I** for information

Just one entry with the severity **E** in the list prevents a regular release of the corresponding delivery request.

An entry is made in the results list of the object list check by raising a specific event for any object set. These events are provided for the object-list-specific interface `IF_EM_OLC_OBJECT`, `IF_EM_OLC_KEY`, and `IF_EM_OLC_STRINGKEY`. The event in question defines the possible action in the results list implicitly. The following events are available:

- `~MISSING`
Like *Add Object*: Adds the object to the piece list.
- `~REJECTED`
Like *Delete Object*: Deletes the object from the piece list.
- `~UNDECIDED`
Like *No action selected*. The user must read the information in the message.

You can specify an optional message text using the parameters of the event. If the parameters are not used, the system variables are read (`SY-MSGID`, `SY-MSGNO`, `SY-MSGVAR1`.... `SY-MSGVAR4`). You can use any message class and its messages. We recommend that you describe the error in detail in the long text. The ABAP command `message` can be used to fill the messages in the system variables in the following variant: `MESSAGE msg/text INTO text` as a call before the event is raised.

Display of the Critical Objects

The results list of every check is displayed as a separate block within the overall result of all object list checks.

The title is the same as the check text specified when the check was activated (in the logon language). The total number of all errors and warnings detected for each function module is also displayed.

If a message text is assigned, all objects or table entries for a message (with the same values for the message variables) are displayed in a sub-block (with the message in question as a title and with any message variables replaced). All objects or table entries without message texts are also displayed within a sub-block.

Within a sub-block, the objects and table entries are sorted by severity and action.

Note

If you want the rows in the results list of a check to be in a different order, you can fill a consecutive number (`I_POS`) when the events are called. This sorts the objects or table entries by the consecutive number.

Here too, consecutive rows for a message (with the same values for the message variables) are displayed in a sub-block (with the message in question as a title and with any message variables replaced).

In addition, all tables entries for a master object (which have the same values for **MASTERTYPE** and **MASTERNAME** (in the parameter structures `I_E071K` and `I_E071K_STR`) within the sub-block for the group text) are grouped and displayed with the master object in question as a header line.

Note

If you only want to display one message text in the results list, you can use a row without assigned object list entry (structure `I_E071` for events of the interface `IF_EM_OLC_OBJECT` is empty) or without assigned key entry (structure `I_E071K` for `IF_EM_OLC_OBJECT_KEY` is empty) or without assigned string key entry (structure `I_E071K_STR` for `IF_EM_OLC_OBJECT_STRINGKEY` is empty).

This message text is displayed as a separate sub-block that consists of only one header line (meaning it does not contain any objects or table entries). However, when the total number of errors and warnings detected by the check module is counted, every message and its severity is respected.

Custom Comments for Handling Entries in the Results List of the Object List Checks

The results list of the object list checks displays all successful and critical objects. In the case of objects with errors, you can accept the check result manually (set it to **OK**). The system requires you enter a reason why this error can be accepted.

You can either enter a text manually or select a predefined text. This text is then added to the comment field where you can then modify it. A predefined text is provided as a default. You can define further texts as follows:

1. Call transaction `SE16` and enter the table name `TRDELVTEXT_TEMPL`.
2. Choose [Create Entries](#).
3. On the next screen, enter a 5-figure number starting with **9** (partner namespace) in the **ID** field for the comment.

4. Enter the text in the [Text](#) field.

Note

The text is not language-dependent and you must create it in a language suitable for all users of Software Delivery Composer.

10.4 Creating Add-On Installation Packages

Use

If you want to deliver the software development for your add-on to customers for the first time, then you need to create an Add-On Installation Package (AOI).

You can also create an Add-On Installation Package if you want to update an add-on that has a small scope. To do so, create an Add-On Installation Package that contains all add-on objects and assign the enhanced attribute REINSTALL_ALLOWED = T to the package in the Software Delivery Assembler. The new Add-On Installation Package can then overwrite the existing add-on with one that it on a lower release. This makes sense for small add-ons as no additional upgrade package is required. However, customers who have modified the objects of their add-on must compare their modifications when they have reinstalled an add-on. For large add-ons you should create [upgrade packages for the add-on delta upgrade \[page 77\]](#) as these only contain the changed and new objects of the add-on.

Procedure

1. Execute the software development in the development system.
2. Import the development into the consolidation or final assembly system.
3. Create the delivery with the tools of the SAP Add-On Assembly Kit (see [Creating Deliveries \[page 57\]](#)).
If you create an Add-On Installation Package to update your add-on, then assign the extended attribute REINSTALL_ALLOWED = T to the package during the registration with the Software Delivery Assembler.

10.5 Making a Delivery Available

To deliver the add-on package, you have the following options:

- Delivered on CD
- Delivered in compressed form on the Internet

The following information and data must be included in the delivery:

- Add-on software

- Required documentation (including installation instructions)

When creating a CD, bear in mind the following points:

- CD structure
- Packing/unpacking the files

For more information about the CD structure and packing and unpacking files, see [CD for the Add-On Delivery \[page 130\]](#).

10.5.1 Template: Installation Guide

Alongside the general product documentation, we recommend that your add-on delivery contains an installation guide. Here you can specify all important information required by your add-on customers to import the packages.

The installation guide should cover the following information:

- Prerequisites
- Preparation
- Execution
- Follow-up

The following text provides the most important information that should be included in an installation guide. You can use this text as a template for your add-on deliveries.

Note

You can also use it to create a guide for other add-on packages (such as upgrade packages or CRTs).

Installation of <add-on><release>

Content

Prerequisites for the installation of <add-on><release>

Preparation of the installation of <add-on><release>

Installation of <add-on><release>

Follow-up of the installation <add-on><release>

Password for the installation of <add-on><release>

Language support

Prerequisites for the Installation of <add-on><release>

- Current kernel, tp , and R3trans
Verify that the current versions of the kernel tp , and R3trans are in your system.
- Current SPAM/SAINT update

Verify that you have imported the latest SPAM/SAINT update into your system. To do this, compare the short text of the last imported SPAM/SAINT update with the short text of the SPAM/SAINT update in SAP Support Portal. If you find a newer version of the SPAM/SAINT update in SAP Support Portal, import it. You can find more information about SPAM/SAINT in SAP Help Portal under <https://help.sap.com/spmanager>.

- Before the installation, read the following SAP Notes or other documentation:

Table 11:

<note title>	<note number>
--------------	---------------

- Check whether the following prerequisites are met in your system:
 - Prerequisite software components (see table ► [System](#) ► [Status](#) ► [Product Version](#) ► [Details](#) ► [Tab: Installierte Softwarekomponentenversionen](#) ►):

Example

Table 12:

SAP_BASIS	740
SAP_BW	740
PI_BASIS	740
or PI_BASIS	730

- Prerequisite support packages:

Example

Table 13:

SAP_BASIS 740	Support package: 06
SAP_ABA 740	Support package: 06

- Additional information about the installation

Example

Table 14:

Space required in the transport directory	Approximately X MB
CD label for the add-on installation	<label number>
Net runtime	Approximately X hours

Preparation of the Installation of <add-on><release>

1. Import the required user languages for all installed components. Perform this language transport before the upgrade or installation of <add-on><release>.
2. Mount the add-on installation CD.
3. Load the packages into your system.
For more information, see the online documentation of Add-On Installation Tool. To do this, choose the help function on the toolbar and then choose ► [Online Documentation](#) ► [Loading Installation Packages](#) ►.

Installation of <add-on><release>

1. Log on to client 000 in your SAP system as a user with SAP_ALL rights. **Do not** log on with the user SAP* or DDIC.
2. Use Add-On Installation Tool (transaction SAINT) to start the installation or upgrade.
For more information, see the online documentation of Add-On Installation Tool. To do this, use the help function in the toolbar.

Follow-Up of the Installation <add-on><release>

Password for the Installation of <add-on><release>

<If you generated a password for the installation, you can specify it here.>

Language Support

Alongside <original language>, <add-on> <add-on release> supports the following languages: <language1>, <language2>.

If you want to install one of these languages, the standard languages of the current SAP release must be installed. Perform the add-on language import as specified in the language transport documentation. You can find the language packages of <add-on> <add-on release> in the directory <language> on the <installation/language CD>.

10.6 Delivering Languages Translated Retroactively

Use

We recommend that your initial add-on delivery contains all languages you require. This means that all languages must be fully [translated \[page 50\]](#). If you have configured the [parameters for exporting language-specific data \[page 43\]](#) correctly, the add-on package is then available in all translated languages.

It may be the case, however, that you need to deliver a language for your add-on after the initial delivery. One example of this is when you use your add-on in your company and your company merges with a company from a country whose language is not available in this add-on. The functions of the add-on are available, but the new company requires its texts in a different language.

Additionally, if you sell your add-on to other companies, non-domestic ones may also be interested in other languages.

You can deliver languages retroactively in the following ways:

- Deliveries using an add-on language package that contains only the translation
- From SAP NetWeaver 7.00: Deliveries using a subsequent support package

Deliveries Using an Add-On Language Package

Prerequisites

- You have already delivered your add-on without the new target language. You have also potentially created support packages for your add-on.
- The underlying SAP components of your add-on support the required target language. Check the availability of the language for the SAP components in question in SAP Service Marketplace under <https://service.sap.com/languages>.
- You have installed the target language in question for the SAP components of your full system in the consolidation system for corrections.
- Language packages are included in separate language CDs delivered with the installation or upgrade. The language transport tool (transaction SMLT) is used to import the language packages.
For information about language transports, see SAP Help Portal under [Language Transport](#)

Activities

1. Prepare the consolidation system for corrections for the translation of the add-on into the new target language
For more information about this, see SAP Help Portal under [Setting Up and Coordinating Translation \(BC-DOC-TTL\)](#).
You must verify that all language-dependent objects in your add-on have been translated. To do this, use the current full add-on piece list as the basis for determining the objects to translate. The full add-on piece list consists of the the component piece list of the add-on and the piece lists of the support packages and conflict resolution transports that have already been created. You can create the full add-on piece list, for example, in Transport Organizer.
If you encounter problems when preparing your systems for the translation, open a problem message on component BC-DOC-TTL.
2. Translate the objects in question into the new target language
For more information about this, see SAP Help Portal under [Translation Tools for Translators \(BC-DOC-TTL\)](#).
3. Create an add-on language packages for the new target language
You create the language package using the language export tool (transaction SMLT_EX).
For information about language exports, see SAP Help Portal under [Language Transport](#)
Note the following points when creating the language package using the language export tool:

General

- When you select the objects, specify the full add-on piece list as the transport request.
- Select all object types.
- Note the special handling of the objects [Terminology](#), [Glossary](#), [Public Holiday Calendar](#), and [Balance Sheets](#). The language guide contains information about this under *Objects with Special Handling*.
- Use only one transport request for the language package. Do not spread the language package across more than one transport request.
- Use the wizard for creating add-ons.
To do this, choose the [Wizard](#) button. If SAP Add-On Assembly Kit is installed in your system, the following query appears: *Use wizard to create add-ons?* Choose Yes.

Note

Do not use the general language export wizard. This wizard appears if you choose the [Wizard](#) button and then answer the query *Use wizard to create add-ons?* with *No*.

- To create a unique language delivery, enter a name for the transport request that matches the add-on delivery requests created in Software Delivery Composer in the *Transport Request* field under *Export Parameters*:

SAPK-*<3-character add-on release name><any two characters>*IN*<namespace>*.**

4. Once the export is complete, the language package is available in the transport directory `EPS/out/` of the consolidation system.

Example

`/usr/sap/trans/EPS/out`

If you do not specify your own name for the language package, it is given the following name:

`<SID><installation number>_<7-figure consecutive number>.PAT`

The transport request is included in the language package. If you have not specified your own transport request name, the system generates a name for the transport request automatically.

5. Test the add-on language package

To test the language package, copy it from the transport subdirectory `EPS/out` in the consolidation system for corrections to the transport directory `EPS/in` in the test system.

If you have not specified the software component on which the language packages based, verify that this component is installed in the system before importing the package.

Also verify that the test system has the same support package level as the consolidation system where you created the language package before you import it. If you import the add-on support packages after you import the language package, it is possible that some of the imported languages are overwritten. This creates a language inconsistency in the test system that can only be resolved by lengthy follow-up work. For more information, see SAP Note [195442](#).

Use the language import tool (transaction SMLT) to import the language package in the test system.

Check that the full add-on is available in the new target language.

6. Migrate the language package to Software Delivery Assembler

Migrate the language package to Software Delivery Assembler in the system where you registered your other packages. This ensures that all packages you create are in the same repository.

The language import tool does not read any attributes specified in Software Delivery Assembler (such as a specific support package level). This means that you do not need to specify these attributes.

7. Deliver the add-on language package

If your tests are successful, you can [provide the language package for delivery \[page 68\]](#).

If the language package is based on a specific add-on support package level, notify your customers that they first need to import all add-on support packages created before the language package was created. Only then can they use the language import tool (transaction SMLT) to import the language package.

8. Follow-up actions

From now on, you need to maintain the add-on in the new target language too, which means you have to add the new language in the [parameter LANGUAGE \[page 43\]](#) in your consolidation system for corrections. From now on, translate all languages in the consolidation system for corrections.

This is how you make sure the support packages you created from now on also contain the new translated target language.

Note

When you deliver your add-on in the target language in question, it is essential that your installation guide advises customers to use the following import order:

- Install the add-on package and all support packages created before the language package was created

- Import the language package
- Import any further support packages that already contain the new language package

Deliveries Using a Subsequent Support Package

1. Create a current full add-on piece list. You can create this, for example, in Transport Organizer.
2. Translate the full add-on piece list into the new target language.
For more information about this, see SAP Help Portal under [Translation Tools for Translators \(BC-DOC-TTL\)](#).
3. Use the language export tool (transaction SMLT_EX) to export the new languages.
4. Use Software Delivery Composer to include the new export in the new support package of the add-on. You must select the new translated target languages explicitly in Software Delivery Composer. To do this, choose [► Delivery Component ► Display/Select Export Languages ►](#).
5. You must use Software Delivery Assembler to set the attribute LANGUAGE_BY_SP with the new translated target languages for the new support package. In imports to a target system, this registers the new delivered languages as installed. The languages specified in the mandatory attribute LANGUAGE must exist and match the selected export languages in Software Delivery Composer. The values for this attribute must be specified using the two-character ISO language key.

Example

LANGUAGE ISO-DEENFRJA

LANGUAGE_BY_SP ISO-FRJA

This package delivers the languages FR (French) and JA (Japanese).

11 Maintenance and Upgrade

You have the following tasks after you deliver an add-on for the first time:

- You must make corrections to any errors in the add-on. You do this by creating add-on support packages. Depending on the underlying SAP release, use either the type CSP (component support packages from SAP Web AS 6.40) or AOP (add-on support packages up to SAP Web AS 6.20).
- If you made modifications to the standard SAP system when developing your add-on, you must restore them if overwritten by a support package delivered by SAP. You do this by creating conflict resolution transports.

➔ Recommendation

We recommend that you only create **non-modifying add-ons**.

- Deliver any further developments to the add-on as an upgrade. You do this by creating add-on upgrade packages.
- Another of your important tasks is to support the add-on in SAP system upgrades. This is because you usually need to update an installed add-on in an upgrade to ensure it can run on the new SAP release. To upgrade an add-on during an SAP upgrade (this is known as an add-on exchange upgrade), create one of the following packages as dictated by the underlying SAP release:

- From SAP NetWeaver 7.0: Add-on exchange package
- SAP Web AS 6.20 and SAP NetWeaver '04: Add-on upgrade package with extended attribute
`CREATE_FULLTASK = T`

The package is included in the correct place in the SAP system upgrade.

If the add-on is compatible enough to run on multiple SAP releases without modification, you can preserve your add-on in the SAP system upgrade to the new SAP release without updating it. The attributes and import conditions of the packages, however, must also describe this new SAP release as a valid target state of the system. If this was not previously the case, the appropriately modified attributes and import conditions can be delivered retroactively using an attribute change package (ACP).

- Before you can convert a system containing one of your add-ons to SAP S/4HANA, you must decide how you want to handle the add-on. There are several options:
 - The add-on can be used on SAP S/4HANA without any modifications.
 - The add-on can be used on SAP S/4HANA but some prework is required before the upgrade.
 - The add-on is no longer relevant in the SAP S/4HANA environment.
 - The add-on is replaced by a new version.

The classification above is required because some basic functions provided by SAP are no longer available in SAP S/4HANA systems. For an overview, see the [simplification database](#).

If the add-on remains available but prework is required before the upgrade, SAP offers a framework where you can provide functions for prechecks and define the compatibility between SAP S/4HANA and the various software components (based on the classification above).

This framework is then used to handle your software automatically during the upgrade. SAP provides support for classification and also if you have questions about creating the precheck or exemptions from the precheck. Also see SAP Note [2308014](#) and the references it contains to additional documents.

11.1 Rules for Add-On Maintenance

Note the following rules when maintaining your add-ons:

- Do not create any new objects.
If you absolutely have to create new objects, first create them in the development system for the lowest required add-on release of your development landscape. Then transport the objects into the development and maintenance systems in the follow-on landscape. This ensures that objects created in maintenance exist in higher releases too and you prevent your add-on from being downgraded.
- Do not make any modifications.
- Only make changes to source code. Do not make any changes to UIs or dictionary objects.
Changes to UIs can make extra training necessary and changes to dictionary objects can, in some circumstances, even lead to loss of data.
- Read the information under [Object List Checks \[page 61\]](#).
- If your add-on contains modifications to a different software component, note also the [rules for developing a modifying add-on \[page 119\]](#).

11.2 Creating Maintenance Packages

An add-on is maintained using add-on support packages and conflict resolution transports (CRTs).

- Correcting errors in the add-on delivery:
Creating add-on support packages (see [Creating Add-On Support Packages \[page 76\]](#)).
- In **modifying** add-ons:
Creating conflict resolution transports (CRTs) (see [Creating Conflict Resolution Transports \[page 120\]](#)).
You must perform the modification adjustment for each SAP support package of the modified software component, if your modifications were overwritten by SAP support packages. You must deliver the adjusted objects in a CRT.
- Development requests that are contained in installation packages should not be delivered again in add-on support packages. The system recognizes this status and offers to remove these requests.

11.2.1 Creating Add-On Support Packages

Use

If you provide corrections for errors in your delivered add-ons, it is advisable to create add-on support packages (CSPs (or AOPs)).

Note also the restrictions that apply to permitted changes. You can find these in [Rules for Add-On Maintenance \[page 76\]](#).

Procedure

To create an add-on support package, proceed as follows:

1. Make the corrections in the maintenance system.
2. Import the corrections into the consolidation/final assembly system for corrections.
3. Create the delivery using the AAK tools (see [Creating a Delivery \[page 57\]](#)).
Create the delivery in the delivery client of the consolidation system for corrections
If you are creating support packages in parallel for your add-on for multiple releases, you must use the extended attribute EQUIVALENT when you register the support package in Software Delivery Assembler. This attribute associates support packages or CRTs with different add-on releases. It is intended to preserve corrections from the add-on support packages in the higher release in SAP system upgrades. For more information about the attribute, EQUIVALENT, see the online documentation in Software Delivery Assembler under [Tab: Extended Attributes](#).

11.3 Creating Add-On Upgrade Packages (Upgrading the Add-On Software)

Use

If you want make developments in a shipped add-on that are not merely corrections to errors, we recommend that you create an add-on upgrade package (AOU). Add-on upgrade packages are used for add-on delta upgrades (that change the add-on release while preserving prerequisite SAP software component version). You use SAP Add-On Installation Tool to import add-on upgrade packages.

➔ Recommendation

If you want to deliver new objects, we recommend that you always create an add-on upgrade package instead of an add-on support package.

i Note

Note that you must also create an add-on upgrade package (AOU) for add-ons based on SAP Web AS 6.20 and 6.40, to ensure that the add-on is updated during an SAP system upgrade. This procedure is described in [Updating Add-Ons in SAP System Upgrades \[page 78\]](#).

Procedure

To create an add-on upgrade package, proceed as follows:

1. Complete your development work in the development system for the add-on release in question.
2. Import the new development work into the consolidation system or final assembly system.

3. Create the delivery using the AAK tools (see [Creating a Delivery \[page 57\]](#)).

If you want to create only add-on upgrade packages to update your add-on (and no new add-on installation packages), you can use the extended attribute `MULTISTEP_INSTALL = T` when you register the add-on upgrade package in Software Delivery Assembler. If this attribute is set, you can import the add-on upgrade package together with a predecessor add-on installation package or predecessor upgrade package from the same component in a single queue. Usually only one add-on installation package or one add-on upgrade package is allowed in a single queue.

11.4 Add-On Behavior in SAP System Upgrades

Use

Note

This section describes the behavior of add-ons in SAP system upgrades. For information about the SAP system upgrade of your add-on development and maintenance landscape, see [Upgrading the Development/Maintenance Landscape \[page 44\]](#).

Whenever SAP publishes a new release (with new prerequisite software versions), you must decide how your add-on reacts.

You have the following options:

- You can update your add-on during the SAP system upgrade.
In this case, you can use SAP Add-On Assembly Kit to define an add-on exchange package (AOX). This ensures that your package is included in the upgrade at the correct place. In the case of add-ons based on SAP Web AS 6.40 and lower, you can do this by defining an add-on upgrade package (AOU) with the extended attribute `CREATE_FULLTASK = T`.

Recommendation

We recommend this option as the default method.

- If your add-on can run on the new SAP release without any modifications, you can preserve your add-on in the SAP system upgrade without updating it. To do this, you must define the appropriate import conditions in the extended attributes in Software Delivery Assembler. For more information, see the online documentation in Software Delivery Assembler.
If the import conditions for the new SAP release or the prerequisite software component version were not yet defined, you can provide your customers with them in an attribute change package (ACP). In this case, the ACP contains the additional import condition for the new SAP release.

Caution

You **cannot** use this option for modifying add-ons.

Remember that you cannot delete your add-on during an SAP system upgrade. For more information, see [End of Maintenance \[page 80\]](#).

- If the add-on upgrade requires preparation or follow-up actions, and you want to make sure that the associated information is read, you can set the attribute UPG_KEY_REQUEST in the extended attributes of the package in Software Delivery Assembler. For more information about this, see the online documentation in Software Delivery Assembler.

Updating Add-Ons in SAP System Upgrades

If your upgrade strategy involves updating the add-on during the SAP system upgrade, you can create an add-on exchange package (AOX) from SAP NetWeaver 7.0.

To create these packages, proceed as follows:

1. Complete your development work in the development system of the new SAP release.
2. Import the new development work into the consolidation system or final assembly system.
3. Create the delivery using the AAK tools (see [Creating a Delivery \[page 57\]](#)).

Note the following points here:

1. In Software Delivery Composer, use the delivery type *Installation/Upgrade* to create a delivery request of the type *Exchange Component Piece* .
The exchange component piece list contains the full add-on. When the exchange component piece list is created, the associated component piece list is flagged automatically for the same add-on release and included. Select the component piece lists of all add-on releases supported as upgrade source releases.
2. Register the exchange component piece list in Software Delivery Assembler. If you choose ► [Delivery Request](#) ► [Register \(by SDA Locally\)](#) ► in Software Delivery Composer, Software Delivery Assembler displays the corresponding tab page automatically:
 - From SAP NetWeaver 7.0: Tab page AOX:
Creates an add-on exchange package
 - Up to and including SAP Web AS 6.40: Tab page AOU:
Creates an add-on upgrade package
This action sets the extended attribute CREATE_FULLTASK to the value T automatically. This ensures that the AOU is accepted as an upgrade package during the SAP system upgrade.

Note the following points when defining the import conditions in Software Delivery Assembler:

- When you define the import conditions, specify the release and support package levels of the software components in the target release in enough detail. For example, you can specify as prerequisites the software components referenced by your add-on using a particular support package level or using other add-ons that reference your add-on.
- In the source release, enter the add-on release of your add-on as a prerequisite and in this way specify the add-on source release for the upgrade. You cannot, however, specify your own support packages or CRTs as a prerequisite for your own add-on source release.
If your add-on also supports further add-on source releases, you can add these by choosing [Alternative Import Conditions](#).

Caution

You cannot use the import conditions to force the installation of a new add-on during an SAP system upgrade.

4. Perform the upgrade to the new release in the test system where you installed your add-on source release.

Up to the phase `UPLOAD_REQUEST` in the program `PREPARE`, place the EPS file of the new add-on exchange package/add-on upgrade package in the transport directory `EPS/in`. The EPS file is the file with the ending `.pat`.

Note


If you [previously package the delivery package in an SAR archive \[page 130\]](#) and place this archive in the directory `/usr/sap/trans`, the EPS file is placed in the directory `EPS/in` automatically when the SAR archive is unpacked. For information about packing and unpacking archives, see [CD for Add-On Deliveries \[page 130\]](#).

During the upgrade, your add-on has the status `UNDECIDED` in the `PREPARE` phase `IS_SELECT`. Select your add-on and confirm it by choosing *OK*. Then choose the option *Upgrade with SAINT Package* for your add-on. The *Add-On Exchange Package/Add-On Upgrade Package* is included in the upgrade in the correct place. If the SAP upgrade tool does not detect your add-on, this could be because the EPS file is not in the transport directory `EPS/in` or because the import conditions for the add-on package are not met.

5. After the SAP system upgrade, verify that your add-on can run on the higher release.
6. Make the add-on exchange package/add-on upgrade package available to your customers and describe the procedure in the `PREPARE` phases `UPLOAD_REQUEST` and `IS_SELECT` (see also: [Providing a Delivery \[page 68\]](#)).

Your customers can now use the add-on exchange package/add-on upgrade package to perform an SAP upgrade to the new release. The add-on is updated in the upgrade and can run on the new release.

Note

Your customers can access more information in the upgrade guide for the release in question, located in SAP Service Marketplace under <https://service.sap.com/instguides> .

11.5 End of Maintenance

It is possible to develop add-ons that can also be uninstalled. Add-ons of this type can be removed from the system after the maintenance phase has ended. Provide instructions about how to delete your add-on from the system in your add-on delivery.

For more information, see [Add-On Uninstallation \[page 81\]](#).

Add-ons that cannot be uninstalled have the following consequences:

- You must verify that the add-on can work for each support package of the SAP release underlying the add-on.
- You must also provide a response to each SAP upgrade. Here, you can either update the add-on at the same time as the upgrade or (in exceptional cases) preserve the add-on without modifying it. For more information, see [Add-On Behavior in SAP System Upgrades \[page 78\]](#).

12 Add-On Uninstallation

It is possible to remove installed add-ons from an SAP system completely with the SAP Add-On Installation Tool (transaction code SAINT).

In order for add-ons to be uninstalled, on the one hand these must fulfill certain criteria and on the other hand they must provide additional information in order to be able to be deleted successfully. The checklists in the following sections contain information for the entire software lifecycle of the add-on, starting with the development phase.

If you are planning on creating an uninstallable add-on, use the checklists in order to understand what you need to take into consideration before and during the creation of an uninstallable add-on.

Caution

After an add-on has been uninstalled, the system must have a consistent state. You must make sure that uninstalling an add-on does not affect the functionality of other add-ons or software components.

Note




At this time it is only possible to uninstall small and simple add-ons. It is not possible to uninstall larger and more complex software components correctly.

The following sections deal with the following questions:

- Can my add-on be uninstalled?
- How can I uninstall an add-on?
- How can I make sure that the add-on is uninstalled without any technical errors?
- What happens with the custom settings after the uninstallation?
- What must I take into consideration in the system landscape?

The following SAP Notes are relevant for creating uninstallable add-ons:

Table 15:

SAP Note	Content
33040 	Options in Exchange Upgrade with Add-On
70228 	Add-Ons: Prerequisites and Upgrade Planning
1883223 	Note on General Add-On Uninstallations

For more information about the technical steps involved in uninstallations, see the SAP Add-On Installation Tool documentation under <http://help.sap.com/spmanager> ► *SAP Add-On Installation Tool* ► *General Description of the Uninstallation Process* ►.

The sections *Phases in the Uninstallation Process* and *Checks in the Uninstallation Process* provide you with tips on how to detect any problems when uninstalling an add-on.

12.1 Checklist: Development Aspects

You must consider the following aspects before and during the development phase of an uninstallable add-on:

Table 16:

Activity	Procedure
Check its relevance to the architecture	<p>You must check the relevance of the add-on in the application architecture. Provide answers to the following questions about the relationship between the add-on and the architecture:</p> <ul style="list-style-type: none">• Is the add-on integrated into the software architecture?• Does the architecture need to be modified when the add-on is deleted?• What effects to these modifications have on the system landscape?• Is the add-on a central component of the product architecture?
Modify table entries	<p>Sometimes tables that are not part of the add-on delivery have to be modified before the add-on can be used. This behavior can be unpredictable, which means that these table entries need to be deleted manually.</p>
Delete activated services in the back-end system	<p>Any services created for the add-on (such as ICF (Internet Connection Framework) services) have to be deleted.</p>
Integration into the system landscape	<p>This can involve components such as SAP NetWeaver Gateway or SAP PI services.</p>
Undo the implementation	<ul style="list-style-type: none">• Check your installation and implementation guides and use installation or configuration automation programs to identify what has been modified or added in the system.• This must include both automated and manual activities. The customer must undo these activities manually.
Implement and activate enhancement spots	<ul style="list-style-type: none">• Does the add-on contain enhancement spots for end users?• Are any enhancement spots planned?
Define namespace conventions	<p>Get an overview of which objects and names are used that were created by the add-on.</p>
Check objects created by the add-on	<p>To identify add-on-specific customizing settings, you must follow naming conventions in the development phase.</p>
Identify generated objects in connected systems	<p>Verify whether there are any generated objects that might no longer exist if the add-on is uninstalled.</p>
Check data in tables outside of the add-on	<p>If you need to modify the add-on, check whether it uses any tables that are part of another add-on.</p>
Check reimports of the deleted add-on	<p>Check what happens if a customer installs a deleted add-on again.</p>
Remove customizing settings	<p>Any data inserted in tables manually must be removed before the add-on is deleted.</p>

Activity	Procedure
Consider uninstallations before dynamic programming	The more complicated your style of programming is, the more difficult it is to delete the add-on. This means it is important that development teams use a simple programming style from the start if they want their add-ons to be uninstallable.
Test whether the deletion was successful.	You must know what to test to verify that the add-on was deleted successfully.

12.2 Checklist: Landscape Aspects

You must consider the following system landscape aspects when creating an uninstallable add-on:

Table 17:

Activity	Procedure
Prepare the add-on uninstallation	The add-on deletion action must be the same as the add-on installation action. For example, product managers must decide whether the add-on is deletable and developers must make this possible.
Reduce customer activities	Ensure that the add-on can be deleted with the least amount of work possible for the customer. This saves time and costs.
Consider reinstallations of the add-on	You must know what happens when customers delete an add-on and then want to install it again.
Access archived data	<ul style="list-style-type: none"> • Check whether any data archives exist. • Check how data archives can be accessed after the add-on is uninstalled. <div> Note If this is not possible, the add-on cannot be deleted. </div>
Make a decision about backup data	<ul style="list-style-type: none"> • Decide what you want to happen with backed up data. • Decide what you want to happen with add-on data in backups after the add-on is deleted.
Modify the guides to handle the add-on	Check whether any guides (or other documentation not part of the system documentation) is affected by the uninstallation. This could include any other documents provided to the customer.
Define a maintenance strategy	<ul style="list-style-type: none"> • If the add-on does continue to exist, you must update this information and test whether it can still be deleted. • If the add-on no longer exists, give your customers enough time to react.

12.3 Checklist: System Aspects

Before uninstalling the add-on you must consider the following aspects with regard to other components in the system where the add-on is installed and other systems that the system is connected to:

Table 18:

Activity	Procedure
Adjust Business Configuration in Other Software Components	It is possible that the configuration in other software components (add-ons) activates the add-on (for example, if a process consists of multiple functions that are in different add-ons). The configuration must be completely adjusted.
Plan Background Processes	<p>No background processes must be running during the deletion. The functions of the add-on must no longer be used productively. All running jobs must be completed before the add-on can be deleted.</p> <ul style="list-style-type: none">• All background processes that affect the add-on must be completed before the add-on can be deleted.• Scheduled background processes must be unscheduled.• Background jobs must be deleted.
Messages Between Add-on and Other Systems	<ul style="list-style-type: none">• Messages that are in the queue and waiting for confirmation must be removed. The add-on sends messages to other processes that are possibly waiting for confirmation. If the add-on is deleted then these messages also need to be removed.• Messages that were sent to the add-on are in the queue. If the add-on is deleted, then there must be a decision made about these messages and they must be located.
Check Cancellation of Workflow	<p>All workflow activities must be canceled from the business side. This means that you must check if workflow processes are still open before the deletion. Workflows that are in process must either be cancelled or executed.</p> <div><p>i Note</p><p>Once the add-on is deleted then you can no longer complete the workflows.</p></div>
Check Configuration for Reuse Options	<p>A system-wide configuration is necessary to be able to reactivate add-ons. Examples for this are RFC connections or communication users.</p> <div><p>i Note</p><p>An add-on can only be deleted when it is no longer in use.</p></div>
Check Dependencies Between Other Systems and the Add-On	Ensure that customer enhancements do not build on a deleted add-on. The add-on must use standard interfaces so that you can control what is affected by the deletion.

Activity	Procedure
Adjust Central Monitoring	If the add-on is connected to the central monitoring by providing data for this? If yes, then this must be set accordingly.
Manage SAP Solution Manager	<ul style="list-style-type: none"> • Inform the system administrator of SAP Solution Manager about the scheduled add-on uninstallation. • Adjust the landscape database.

12.4 Checklist: Test Aspects

You must take the following aspects into consideration with regard to tests on an uninstalleable add-on:

Table 19:

Activity	Procedure
Validate the uninstallation process	Ensure that the process of the add-on uninstallation is the opposite of the process of the add-on. For example, product management must confirm the decision that the add-on is to be deleted and development must check that the work to be executed and the deletion function correctly.
Test whether the deletion was successful.	Ensure that you know what you need to test in order to determine if the add-on deletion was successful.
Check the new installation of the deleted add-on	Check what happens if a customer wants to install the deleted add-on again.
Document the full add-on uninstallation process	Document all steps that you executed during the deletion in an add-on uninstallation guide.

12.5 Checking Add-Ons for Uninstallability

Before you uninstall an add-on, you must make sure that the add-on can actually be deleted without errors. How you check this is described below. The add-on is considered to be uninstalleable if it has passed the following check steps.

Procedure

1. If the add-on that you want to delete needs to be made uninstalleable post delivery, you generate an attribute change package (ACP).

For the system to recognize the add-on as uninstalleable, the attributes `DEINSTALL_ALLOWED` and, if required, `DEINSTALL_PLUGIN` must be set for the installation package. If you want to make an add-on uninstalleable post delivery, a corresponding delivery for attributes for the software component (an ACP) must be created. If

an ACP already exists, a new version must be created containing the named attributes. If the add-on that you want to delete was marked as uninstalleable when it was created, the attributes are already set for the installation package, and you do not need to create an ACP.

For more information about post-delivering attributes, see the section [Errors in the Delivery \[page 59\]](#).

2. Set up a test system.

To test the uninstallation, use a suitable test system. If necessary, set up a new system for this purpose.

3. Create a test plan.

The test plan should make sure that functions used in the environment of the add-on continue to run without errors after the add-on has been uninstalled, and that no functions have been removed that are still required (for example, by dynamic calls).

4. Import the current SPAM/SAINT update into your test system.

Download the most recent SPAM/SAINT version from SAP Support Portal at <https://support.sap.com/swdc> and install this version using Support Package Manager (transaction SPAM).

For more information about Support Package Manager, see <http://help.sap.com/spmanager>.

5. Install the add-on software component that you want to delete in your test system along with any other associated packages, for example, support packages.

If the add-on that you want to delete was marked as uninstalleable post delivery and an ACP was created, import the ACP as well. Generate realistic application data so that runtime data can be taken into consideration as well in the uninstallation.

6. Start a test run for the uninstallation.

- a. Use transaction SAINT to call the SAP Add-On Installation Tool.
- b. Make sure that the add-on that you want to delete is listed on the [Uninstallable Components](#) tab.

Note

If you can't see the [Uninstallable Components](#) tab, or the component you want to delete is not in the list, either the ACP is missing or the uninstallation attributes have not been set for your component. In this case, first generate an ACP.

- c. Under **Extras > Settings**, select the scenario [Test](#) on the [Import Queue](#) tab and confirm your entry.
- d. Mark the add-on that you want to uninstall and start the uninstallation process.

Since this process is running in a test scenario, all the checks are done, but the add-on is not actually uninstalled.

If errors occur in the COLLECT_OBJLIST phase, please contact SAP Support for component BC-CTS-AAK. Let us know in your message which object is affected.

If an add-on cannot be uninstalled because other add-ons depend on it, uninstallation is only possible if the dependent add-ons are uninstalled as well. These add-ons may need to be made uninstalleable first. Then perform the uninstallation test for the add-on and its dependent add-ons. As of SPAM/SAINT version 59 it is possible to uninstall several add-ons at the same time.

If there are no errors in the test run, you can move on to the next step.

7. Uninstall the add-on in the test system.

- a. Use transaction SAINT to call the SAP Add-On Installation Tool.

- b. Under **Extras** **Settings**, select the scenario *Standard* on the *Import Queue* tab and confirm your entry.
- c. Mark the software component that you want to uninstall on the *Uninstallable Components* tab and start the uninstallation.
- d. To complete the uninstallation, confirm the queue.

If there are no errors, you have successfully deleted the software component from the test system. It is no longer listed as an installed component.

8. Go through the [Test Plan \[page 86\]](#) that you created to check your system after uninstallation.

- **No Errors in Test Run**

If no errors occur during the test run, the add-on is currently uninstallable.

➔ Recommendation

The check result is a snapshot, which you do not want to be endangered by subsequent deliveries (for example, support packages). To be on the safe side, you should always perform an object list check for uninstallation for any subsequent deliveries. We also recommend repeating the check described here, because it cannot be guaranteed that the add-on is permanently uninstallable.

Inform your customers about the uninstallability of the add-on. If the add-on has already been delivered, provide the ACP to your customer as well.

i Note

The above process only covers the technical deletion. Any preparatory steps (for example, data backup), descriptions of dependencies, or resolutions of uses must be described in the documentation for your add-on. If an ACP is required, this should also be mentioned in the documentation.

- **Errors in Test Run**

If errors occur during the test run that cannot be resolved or ignored, the add-on is not uninstallable. In this case, the attributes `DEINSTALL_ALLOWED` and `DEINSTALL_PLUGIN` must be deleted for the installation of the software component. If an ACP existed before the test, this must be changed accordingly and delivered again.

12.6 Attributes for Uninstallations

The extended attributes `DEINSTALL_ALLOWED` and `DEINSTALL_PLUGIN` in Software Delivery Assembler are relevant for uninstallations.

- **DEINSTALL_ALLOWED**

The attribute `DEINSTALL_ALLOWED` specifies whether the add-on can be deleted. If this attribute is set, the add-on is flagged as uninstallable. This attribute can be set only for deliveries of add-ons for the package types AOI, AOU, and AOX or post-deliveries using package type ACP. The attribute has the number of an SAP Note containing general information about the uninstallation. Since you cannot create your own SAP Notes, the system uses the default [1883223](#) when Add-On Assembly Kit is used.

- **DEINSTALL_PLUGIN**

You can specify the name of an interface class in the attribute `DEINSTALL_PLUGIN` that supports the uninstallation process using add-on-specific functions.

For more information, see [Plug-In Interface for Add-Ons \[page 88\]](#).

For examples of how to set attributes, see [Examples: Attributes for Add-On Uninstallations \[page 128\]](#).

12.7 Plug-In Interface for Add-Ons

In some cases, add-ons create dynamic objects of their own. To delete these during the uninstallation, the add-on must inform the uninstallation framework about such objects. It can also happen that an add-on can only be uninstalled under certain conditions which have to be checked by the add-on itself.

An add-on can support its uninstallation using its own methods. You can define a check with which you can check if an add-on can be deleted before the deletion.

In addition to this, the add-on can provide a list of objects that were created dynamically and which need to be deleted.

The add-on can define a class that is called during the deletion process. The name of this class is provided in add-on attribute `DEINSTALL_PLUGIN`. The value of this attribute is the name of the implementing plug-in class. If this attribute is set, the uninstallation checks if the class exists. If it does not exist, the add-on uninstallation is cancelled with a corresponding message. If the class exists, the plug-in is instantiated with the static method `GET_REFERENCE`.

Note

If no class is called during the uninstallation, you must not set the `DEINSTALL_PLUGIN` attribute.

If the instantiation was successful, the system checks if the interface methods exist and calls them if they do. (At least instance method `GET_REFERENCE` must exist for the plug-in class.) This takes place in the following phases in the SAP Add-On installation tool:

- `CHECK_SWCV`: This phase checks if the add-on can be deleted. It calls method `CHECK_PRECONDITIONS` in the plug-in interface. If the method interface returns `EV_CHECK_RESULT = space`, the deletion is deleted. It must also display the corresponding error messages using `ET_MESSAGES`.
- `COLLECT_OBJLIST`: This phase collects all objects that have to be deleted by calling method `GET_CREATED_OBJECTS` and puts all objects in to the deletion task.

Methods of Add-On Plug-In Class

In order for the implementation to work, the interface must have the following properties:

- Static method `GET_REFERENCE` (mandatory)
If possible, this method instantiates the plug-in interface and returns an instance of the plug-in class. The import parameters can be used to select the correct plug-in instance. These can also be forwarded to the

plug-in instance so that the information can be used in the plug-in methods. The interface has the following properties:

- Important parameter (mandatory):
 - `IV_COMPONENT - TYPE DLVUNIT`: Name of the add-on to be uninstalled
 - `IV_RELEASE - TYPE SAPRELEASE`: Release of the add-on to be uninstalled
- Export parameter:
 - `EO_PLUGIN - TYPE REF TO OBJECT`: Plug-in instance
 - `ET_MESSAGES - TYPE SPROT_U_TAB`: Table with log entries (can be empty)
 - `EV_RC - TYPE INT4`: Was the method call successful?
Values for `EV_RC`:
 - `EV_RC = 0`: Method call was successful
 - `0 < EV_RC <= 4`: Method call completed with warnings but the uninstallation can be continued
 - `EV_RC > 4`: Method call ended with errors, uninstallation procedure was completed
- Instance method `CHECK_PRECONDITIONS` (optional)
You can use this method to check if it is possible to uninstall the add-on. For example, if all prework was completed or if a function was only used in a specific mode. The interface has the following properties:
 - Export parameter:
 - `EV_CHECK_RESULT - TYPE FLAG`: 'X' = uninstallation possible, ' ' = not possible
 - `ET_MESSAGES - TYPE SPROT_U_TAB`: Table with messages (can be empty)
 - `EV_RC - TYPE INT4`: Was the method call successful?
You find information about the values in the previous instance method.
- Instance method `GET_CREATED_OBJECTS` (optional)
This method can, if necessary, be used to collect objects and table keys and return them to the add-on uninstallation framework. This makes sense for objects that cannot be found by the standard function for object collection contained in the add-on uninstallation framework, for example, because they are created while the add-on is used. The result is still checked in the following uninstallation phases. The interface has the following properties:
 - Export parameter:
 - `ET_E071 - TYPE TR OBJECTS`: Table with transport objects to be deleted
 - `ET_E071K - TYPE TR KEYS`: Table with table keys (as required)
 - `ET_MESSAGES - TYPE SPROT_U_TAB`: Table with messages (can be empty)
 - `EV_RC - TYPE INT4`: Was the method call successful?
You find information about the values in the previous instance method.

Test the implementation of the interfaces. Development errors can cause add-ons to be incompletely uninstalled and other unwanted side effects.

12.8 Handling Object Types

The following table contains a consolidated list of objects that can be deleted using SPAM/SAINT version 0058 (or higher).

During the add-on uninstallation it is only possible to delete complete objects (R3TR *). If the consolidated object list of the deletion package contains subobjects (LIMU* objects) or language parts of objects (LANG * objects),

these parts are replaced by their complete objects (for example, LIMU FUNC xyz is replaced by R3TR FUGR abc and LANG DOMA xyz is replaced by R3TR DOMA xyz). The tests are executed for the complete objects.

i Note

The description of the object types in the following table is only available in English.

Table 20: Deletable objects

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
CORR	MERG	*	Informational entry regarding history of object list	Retained	53
LIMU	COMM	*	Object list delivered by add-on	Retained	53
R3TR	ACGR	*	Role with authorizations and menus	Deleted	53
R3TR	ACGT	*	Role - User assignment	Retained	59
R3TR	ACID	*	Checkpoint group	Deleted	54
R3TR	ACLA	*	Archiving class	Deleted	57
R3TR	AOBJ	*	Archiving object	Deleted	57
R3TR	AQBG	*	ABAP query: User group	Deleted	57
R3TR	AQQU	*	ABAP query: Query	Deleted	57
R3TR	AQQV	*	ABAP query: Query variant	Deleted	57
R3TR	AQSG	*	ABAP query: Functional area	Deleted	57
R3TR	ASFC	*	SAP AS: Field catalog	Deleted	58
R3TR	ASIS	*	SAP AS: Archiving Information Structure	Deleted	58
R3TR	AUTH	*	Authorization Check Fields	Deleted	54
R3TR	AVAR	*	Activation Variants for Assertions and Breakpoints	Deleted	58
R3TR	AVAS	*	Classification of development objects	Deleted	53
R3TR	AVAS	00505695007C1E D288E822748DD7 E895	Classification	Retained	55
R3TR	AVAS	00505695007C1E D288E822748DD7 E895	Classification	Retained	55
R3TR	AXTA	*	Application Group	Deleted	60
R3TR	AXTB	*	Extensible business objects	Deleted	60
R3TR	AXTP	*	Extensible place	Deleted	60
R3TR	BMFR	*	Application Component Definition	Deleted	54

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	BOTY	*	Object Type for General Object Number	Deleted	57
R3TR	BSVI	*	Status management: System status	Deleted	57
R3TR	BSVO	*	Status management: Object type	Deleted	57
R3TR	BSVS	*	Status management: Status profile	Deleted	57
R3TR	BSVV	*	Status management: Process	Deleted	57
R3TR	CDAT	/IWBEP/VC_MGW_SG	Mobile Gateway: Display Services and Models	Deleted	53
R3TR	CDAT	APPL_LOG	Definition of application specific logs	Retained	53
R3TR	CDAT	ARCHIVE	Archiving object maintenance	Deleted	57
R3TR	CDAT	BUPATBZ1AR	BDT: Event Function Modules	Deleted	58
R3TR	CDAT	COMV_PARTNER_PDP	Maintenance Cluster for Maintaining Partner Procedures	Retained	58
R3TR	CDAT	COMV_TEXT_CUST	Customizing Text Processing	Retained	60
R3TR	CDAT	CRMVC_GIL_APP_DEF	View Cluster for Maintaining Basic Settings for Generic IL	Deleted	60
R3TR	CDAT	CRMVC_GIL_SO_DEF	Maint.View Cluster for Defining Simple Objects in Generic IL	Deleted	60
R3TR	CDAT	CRMV_PROCESS_MA	Maintenance of CRM transaction types and dependent tables	Retained	58
R3TR	CDAT	CUSTA2EVNT	System/User status events	Deleted	57
R3TR	CDAT	FILENAME	Logical file name maintenance	Deleted	55
R3TR	CDAT	LRM_CUST_BS	IRM: Tables for the Business Suite	Deleted	58
R3TR	CDAT	LRM_CUST_DEV	IRM Customizing (Developer View)	Deleted	58
R3TR	CDAT	MB_DELIV	Create delivery in Inventory Management	Retained	57
R3TR	CDAT	SWECDCLUST	Cluster for Maintaining Event Linkage with Change Documents	Retained	57
R3TR	CDAT	SWFVCT	Visualization of Tasks and Objects	Retained	60
R3TR	CDAT	T599C	HR Report Categories	Deleted	59
R3TR	CDAT	T778A	Evaluation paths	Deleted	57
R3TR	CDAT	T778T	Info Types	Deleted	60
R3TR	CDAT	T778V	Links	Deleted	57
R3TR	CDAT	T77AP	Aspects	Deleted	57

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	CDAT	T77FRAMEWORK	Hierarchy Framework	Deleted	57
R3TR	CDAT	T77OBJMAN	Object Manager	Retained	57
R3TR	CDAT	VC_FPB_PER-SAPPL	Personalization: Maintenance of Hierarchy	Deleted	57
R3TR	CDAT	VC_PERSMAIN	Personalization: Maintenance of Dialogs	Deleted	60
R3TR	CDAT	VC_T5961	Areas	Retained	60
R3TR	CDAT	VC_TBANK_CD_D V	Change Document Tool: Settings by the Application Developer	Deleted	58
R3TR	CDAT	VC_TBANK_PP_A PPL	Settings for Parallel Processing Framework	Deleted	53
R3TR	CDAT	VED_TMSG2	Process codes, inbound	Deleted	60
R3TR	CDAT	VERSION_K	CO Versions (Controlling Area Initial Screen)	Retained	59
R3TR	CDAT	V_CLCNVIM-GANDSTATE	PCL: Maintenance of CNVMBTIMG and related state entries	Deleted	60
R3TR	CDAT	V_CLCNVMBTPROCTY	MBT PCL Maintenance of Process Type	Deleted	60
R3TR	CDAT	V_CNVCLMBTACT	MBT PCL Activity Maintenance cluster	Deleted	60
R3TR	CDDH	*	Development Classes for Table Entries: Granularity	Deleted	58
R3TR	CHDO	*	Change Document Object	Deleted	55
R3TR	CINS	*	Correction instruction downloaded as part of SAP Note (not part of deliveries)	Retained	53
R3TR	CLAS	*	ABAP Objects Class	Deleted	53
R3TR	CLAS	CL_DMC_TYPE_INFORMATION_LIST	Specific DMIS-related ABAP Objects Class	Retained	57
R3TR	CUS0	*	Customizing IMG Activity	Deleted	53
R3TR	CUS1	*	Customizing Transactions	Deleted	53
R3TR	CUS2	*	Customizing Attributes	Deleted	53
R3TR	DEVC	*	Package (formerly known as Development Class)	Deleted	53
R3TR	DEVC	S_DMCM	Package	Retained	55
R3TR	DEVC	S_DMCR	Package	Retained	55
R3TR	DIAL	*	Dialog Module	Deleted	60

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	DOCT	*	General Text	Deleted	53
R3TR	DOCV	*	Documentation (Independent)	Deleted	53
R3TR	DOMA	*	Domain	Deleted	53
R3TR	DSAD	*	DataSource Application Component (Delivered Version)	Deleted	57
R3TR	DSYS	*	Chapter of a Book Structure	Deleted	53
R3TR	DTL	*	Data Element	Deleted	53
R3TR	ECAT	*	eCATT Test Script	Deleted	53
R3TR	ECSD	*	eCATT System Data Container	Deleted	53
R3TR	ECTC	*	eCATT Test Configuration	Deleted	53
R3TR	ECTD	*	eCATT Test Data Container	Deleted	53
R3TR	ENHC	*	Composite Enhancement Implementation	Deleted	55
R3TR	ENHO	*	Enhancement Implementation	Deleted	53
R3TR	ENHS	*	Enhancement Spot	Deleted	53
R3TR	ENQU	*	Lock Object	Deleted	53
R3TR	ENSC	*	Composite Enhancement Spot	Deleted	53
R3TR	FORM	*	SAPscript Form	Deleted	58
R3TR	FUGR	*	Function Group	Deleted	53
R3TR	HLPF	*	Link DS/screen field	Deleted	55
R3TR	HRNR	*	Organizational Management and Workflow: Prefix Numbers	Deleted	57
R3TR	HRSF	*	HRSFI: Field Set for Data Import	Deleted	60
R3TR	IAML	*	Language-Dependent IAC Binary Data	Deleted	60
R3TR	IAMU	*	Language-Independent IAC Binary Data	Deleted	60
R3TR	IARP	*	Parameters of IAC Language Resource	Deleted	60
R3TR	IASP	*	Parameters of an IAC service	Deleted	60
R3TR	IATL	*	Language-Dependent IAC Templates	Deleted	60
R3TR	IATU	*	Language-Independent IAC Templates	Deleted	60
R3TR	IDOC	*	Basic type	Deleted	59
R3TR	INTF	*	Interface (ABAP Objects)	Deleted	53

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	IWMO	*	SAP Gateway Business Suite Enablement - Model	Deleted	53
R3TR	IWOM	*	SAP Gateway: Model Metadata	Deleted	57
R3TR	IWPR	*	SAP Gateway BSE - Service Builder Project	Deleted	53
R3TR	IWSG	*	SAP Gateway: Service Groups Metadata	Deleted	57
R3TR	IWSV	*	SAP Gateway Business Suite Enablement - Service	Deleted	53
R3TR	LODC	*	HRDSYS: Logical information object client-dep. table C	Deleted	60
R3TR	LODC	B2A__AUTO-BAAAAAAAAAAAAAAA05_OVVV	HRDSYS: Logical information object client-dep. table C	Retained	60
R3TR	LODE	*	HRDSYS: Logical information object client-dep. table E	Deleted	57
R3TR	LPDS	*	Launchpad Short Texts	Deleted	54
R3TR	LRCC	*	LRepository cross-client content	Deleted	60
R3TR	LRCD	*	LRepository client-dependent content	Deleted	60
R3TR	MCID	*	Matchcode ID	Deleted	58
R3TR	MCOB	*	Matchcode Object	Deleted	58
R3TR	MSAG	*	Message Class	Deleted	53
R3TR	NOTE	*	SAP Note downloaded to the system (not part of deliveries)	Retained	53
R3TR	NROB	*	Number Range Objects	Deleted	55
R3TR	NSPC	*	Namespace in R/3 Repository	Deleted	53
R3TR	NWSE	*	Extractor Object	Deleted	58
R3TR	OSOD	*	DataSource (Delivered Version)	Deleted	57
R3TR	PARA	*	SPA/GPA Parameters	Deleted	55
R3TR	PDAC	*	Standard Rule	Deleted	58
R3TR	PDTG	*	Task Group	Deleted	57
R3TR	PDTS	*	Standard Task	Deleted	57
R3TR	PDWS	*	Workflow Templates	Deleted	57
R3TR	PERS	*	Personalization Object	Deleted	57
R3TR	PINF	*	Package Interface	Deleted	53

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	PROG	*	Program	Deleted	53
R3TR	PROG	DMC_UMS-CHLUESSELUNG	Program	Retained	55
R3TR	PROG	DMC_UNIVERSAL-STARTREPORT	Program	Retained	55
R3TR	SAMC	*	ABAP Messaging Channels	Deleted	58
R3TR	SAMT	*	Message Types for ABAP Messaging Channels	Deleted	58
R3TR	SAPC	*	Object type for APC	Deleted	58
R3TR	SBXL	*	Logical information object for BDS: AEW ABAP tools	Deleted	53
R3TR	SBXP	*	Physical information object for BDS	Deleted	53
R3TR	SBYL	*	Logical information object for BDS	Deleted	53
R3TR	SBYP	*	Logical information object for BDS	Deleted	53
R3TR	SCAT	*	Test case	Deleted	53
R3TR	SCP1	*	BC Set or Customizing Profile	Deleted	53
R3TR	SCVI	*	Screen variants	Deleted	53
R3TR	SFB1	*	Business function set	Deleted	55
R3TR	SFB2	*	Business Function	Deleted	55
R3TR	SFBF	*	Business Function + Assignment	Deleted	55
R3TR	SFBS	*	Business Function Set + Assignment	Deleted	55
R3TR	SFPF	*	Form Object: Form	Deleted	53
R3TR	SFPI	*	Form Object: Interface	Deleted	53
R3TR	SFRN	*	Logical object - information object: Release note	Deleted	53
R3TR	SFSW	*	Switch + Assignment of Objects to the Switch	Deleted	55
R3TR	SHI3	*	General structure storage: Definition of a structure	Deleted	53
R3TR	SHI5	*	Gen. hierarchy storage extension name	Deleted	53
R3TR	SHI6	*	Gen. structure repos.: Extension ID / structure assignment	Deleted	53
R3TR	SHI7	*	General hierarchy store node attribute	Deleted	55

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	SHI8	*	SFW Switch Assignment in Hierarchy Tool	Deleted	58
R3TR	SHI9	*	Gen. Structure Repository: Enhancement ID Sequence	Deleted	57
R3TR	SHLP	*	Search Help	Deleted	53
R3TR	SHMA	*	Shared Objects: Defined Area Attributes	Deleted	53
R3TR	SICF	*	ICF Service	Deleted	53
R3TR	SIM1	*	IMG attributes	Deleted	55
R3TR	SIMH	*	IMG Structure Repository: Nodes	Deleted	60
R3TR	SIRS	*	Customizing for Content Repositories	Deleted	57
R3TR	SMIM	*	Info Object from the MIME Repository	Deleted	53
R3TR	SOBJ	*	Business object types	Deleted	55
R3TR	SOTR	*	All Concepts (OTR) of a Package - Short Texts	Deleted	53
R3TR	SOTS	*	All Concepts (OTR) of a Package - Long Texts	Deleted	55
R3TR	SPRX	*	Proxy Object	Deleted	54
R3TR	SRFC	*	RFC service definition	Deleted	55
R3TR	SRTR	*	Report tree	Deleted	58
R3TR	SSFO	*	SAP Smart Form	Deleted	53
R3TR	SSST	*	SAP Smart Style	Deleted	53
R3TR	STVI	*	Transaction variants (cross-client)	Deleted	57
R3TR	STYL	*	SAPscript Style	Deleted	60
R3TR	SUSC	*	Authorization object class	Deleted	54
R3TR	SUSH	*	Assignment: Service --> Authorization Objects	Deleted	53
R3TR	SUSO	*	Authorization object	Deleted	54
R3TR	SUSP	*	Templates for authorization profiles (Profile Generator)	Deleted	55
R3TR	SXCI	*	Business Add-Ins - Implementations	Deleted	53
R3TR	SXSD	*	Business Add-Ins - Definitions	Deleted	53
R3TR	TABL	*	Table	Deleted	53

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	TABU	/BA1/FO_FIELDS	Field of Analyzer(Char. and Key Fig. Service not Known)	Deleted	57
R3TR	TABU	/BA1/P3_TA_APADS	Application Assignment: Derivation	Deleted	57
R3TR	TABU	/BA1/P3_TA_APFF	Define Fixed Fields for Application	Deleted	57
R3TR	TABU	/BA1/P3_TA_APP	Application Table for Costing	Deleted	57
R3TR	TABU	/BA1/P3_TA_APPT	Application Table for Costing, TEXT	Deleted	57
R3TR	TABU	/BA1/P3_TA_CTGL	Catalog List (Costing)	Deleted	57
R3TR	TABU	/BA1/P3_TA_EC	Environment Catalog (Costing)	Deleted	57
R3TR	TABU	/BA1/P3_TA_FFMS	PA: Function Modules of Function Catalog; with Namespace	Deleted	57
R3TR	TABU	/BA1/P3_TA_FFMST	PA: Function Modules of Function Catalog; with Namespace Text	Deleted	57
R3TR	TABU	/BA1/P3_TA_SMSGC	Processing Control for Application	Deleted	57
R3TR	TABU	/BA1/P3_TA_STY	Permitted Step Categories	Deleted	57
R3TR	TABU	/BA1/P3_TA_STYT	Text Table for Permitted Step Categories	Deleted	57
R3TR	TABU	/BA1/RO_FVS_APP	User of the Fair Value Server	Deleted	57
R3TR	TABU	/BA1/RO_FVS_APPT	User of the Fair Value Server	Deleted	57
R3TR	TABU	/BA1/TF3_UOID_C	Registration of Component-Specific Connector Implementations	Deleted	57
R3TR	TABU	/BA1/TRO_APPCTN	Applications Registered in the FV/AC Server	Deleted	57
R3TR	TABU	/BA1/XX_CMP	Table of Components in the Bank Analyzer	Deleted	57
R3TR	TABU	/BA1/XX_CMPT	Text Table of Components in the Bank Analyzer	Deleted	57
R3TR	TABU	/SAPPO/S_BPROC	SAP Business Processes	Deleted	58
R3TR	TABU	/SAPPO/S_BPROC_T	Text: SAP Business Processes	Deleted	58
R3TR	TABU	/UI2/SYSTEMA-LIAS	System alias for generation of FIORI catalogs based on back-end catalogs	Retained	60

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	TABU	/UI5/TREP_FILES	UI5 mapping table for row files with text entries	Deleted	57
R3TR	TABU	/UI5/TREP_TEXT	UI5 mapping table for texts from row files	Deleted	57
R3TR	TABU	/UI5/TREP_TEXT_T	UI5 table for translated texts from row files	Deleted	57
R3TR	TABU	AGR_TIMEB	Time Stamp for Role (Profile Generation)	Retained	54
R3TR	TABU	ALTDESCR	Alert: Method Description - Language-Dependent	Deleted	55
R3TR	TABU	ALTOOLCHEK	Alert: Tools that have been checked as usable	Deleted	55
R3TR	TABU	ALTOOLEXEC	Alert: Tool definition (executable, dispatcher, and so on)	Deleted	55
R3TR	TABU	APB_LAUNCH-PADT	Descriptions for APB_LAUNCHPAD	Deleted	54
R3TR	TABU	APB_LAUNCH-PAD_V	Launchpad with versions	Deleted	54
R3TR	TABU	APB_LPD_CONTROL	Launchpad: Role and Instance	Deleted	54
R3TR	TABU	APB_LPD_OTR_KEYS	OTR keys for Launchpad texts	Deleted	54
R3TR	TABU	APB_LPD_VERSIONS	Launchpad: Role and Instance	Deleted	54
R3TR	TABU	ARBFND_I_APPMSGT	Control tables of the Ariba integration layer	Deleted	54
R3TR	TABU	ARBFND_I_DCXML_D	Control tables of the Ariba integration layer	Deleted	54
R3TR	TABU	ARBFND_I_DCXML_H	Control tables of the Ariba integration layer	Deleted	54
R3TR	TABU	ARBFND_I_DCXML_V	Control tables of the Ariba integration layer	Deleted	55
R3TR	TABU	ARBFND_I_IFEH_AP	Control tables of the Ariba integration layer	Deleted	55
R3TR	TABU	ARBFND_I_PROC	Control tables of the Ariba integration layer	Deleted	54
R3TR	TABU	ARBFND_I_TRFM	Control tables of the Ariba integration layer	Deleted	54
R3TR	TABU	BALOBJ	Application log: Objects	Retained	55
R3TR	TABU	BALOBJT	Application Log: Object Texts	Retained	55

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	TABU	BALSUB	Application log: Subobjects	Retained	55
R3TR	TABU	BALSUBT	Application Log: Subobject Texts	Retained	55
R3TR	TABU	BDS_LOCL	BDS: Assignment of Application Classes to KPro Doc. Classes	Deleted	55
R3TR	TABU	CNVMBTACTIVITY	Unique ID for all activities, information about execution	Deleted	60
R3TR	TABU	CNVMBTACTPAR-AMS	Parameters for Activities	Deleted	60
R3TR	TABU	CNVMBTACTTS-REF	MBT PCL Reference between Activity and TS process	Deleted	60
R3TR	TABU	CNVMBTCOPYC	Copy Control Data for TDMS & CMIS packages	Deleted	60
R3TR	TABU	CNVMBTIMG	Process plan, order of activities to be executed	Deleted	60
R3TR	TABU	CNVMBTIMG-HEAD	IMG header table - contains IMG change information	Deleted	60
R3TR	TABU	CNVMBTPACK	Package information/creator/creation date/scenario	Deleted	60
R3TR	TABU	CNVMBTPACKDP	Info about dependent packages and versions	Deleted	60
R3TR	TABU	CNVMBTPROC-TYPE	Master data for MBT/PCL procedure types	Deleted	60
R3TR	TABU	CNVMBTRESE-TACT	Reset activities with reference variant	Deleted	60
R3TR	TABU	CNVMBTRESE-TVAR	Variants for the reset activities	Deleted	60
R3TR	TABU	CNVMBTRESE-TVAR_M	Master table for creation of activity status reset variant	Deleted	60
R3TR	TABU	CNVMBTSPEZTAB	Specific rules for copy/load scenario	Deleted	60
R3TR	TABU	CNVMBTSTATE	Status table for execution of all activities and history	Deleted	60
R3TR	TABU	CNVMBTTSDEF	MBT PCL Troubleshooting process definition	Deleted	60
R3TR	TABU	CNVMBTTSH	MBT PCL Troubleshooting processes header	Deleted	60
R3TR	TABU	CRMC_Q10_FIELD S	Field names for One Order Search	Deleted	58

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	TABU	CRMC_REPDY	Control Table for Dyn. Access (CL_CRM_REPORT_ACC_DY-NAMIC)	Deleted	58
R3TR	TABU	CSL_EOME	CSL: Mapping Element EDTY<-> EODD	Deleted	58
R3TR	TABU	CSL_EOMP	CSL: Mapping Ddic Block Object <-> Block Object Type	Deleted	58
R3TR	TABU	CSL_EOTY	CSL: Cross-Component and Cross-Release Object Type	Deleted	58
R3TR	TABU	CUSAH	IMG attribute table: SAP data	Deleted	55
R3TR	TABU	CUS_ACTEXT	Customizing Activity - Assigned Enhancement Object	Deleted	60
R3TR	TABU	CUS_ACTH	Customizing Activity - Header Data	Deleted	60
R3TR	TABU	CUS_ACTOBJ	Customizing Activity - Object List	Deleted	60
R3TR	TABU	CUS_ACTT	Customizing Activity Text Table	Deleted	60
R3TR	TABU	CUS_ATRH	Customizing Attributes - Header Data	Deleted	60
R3TR	TABU	CUS_ATRT	Text Table for Customizing Attributes	Deleted	60
R3TR	TABU	CUS_IMGACH	IMG Activities	Deleted	60
R3TR	TABU	CUS_IMGACT	Text Table for IMG Activity	Deleted	60
R3TR	TABU	CVERS_REF	Reference Table for CVERS Entries	Deleted	53
R3TR	TABU	DSYAD	Structures: Display Structure Without Text	Deleted	55
R3TR	TABU	DSYAH	Structures: Header Data	Deleted	55
R3TR	TABU	DSYAI	Structures: Structure Short Text	Deleted	55
R3TR	TABU	DSYAS	Structures: Maintenance Structure Without Text	Deleted	55
R3TR	TABU	DSYAT	Structures: Texts for Maintenance Structure	Deleted	55
R3TR	TABU	DSYAV	Structures: Directory of Views on Books	Deleted	55
R3TR	TABU	ECHS_DEFLTRESOL	SAP Default Resolution Strategy	Deleted	54
R3TR	TABU	ECHS_PP_PROCESS	SAP Postprocessing (Bijection ECH<-> PPO)	Deleted	54
R3TR	TABU	ECHS_PROCESSSES	SAP Business Process and Assgmt of Action Persistence Notif.	Deleted	58

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	TABU	ECHS_PROCESSES_T	Text - SAP Business Process	Deleted	58
R3TR	TABU	EDBAS	Basic types	Deleted	60
R3TR	TABU	EDBAST	Short Description of Basic Type	Deleted	60
R3TR	TABU	EDIFCT	IDoc: Assignment of FM to log. message and IDoc type	Deleted	59
R3TR	TABU	EDIMSG	Output Types and Assignment to IDoc Types	Deleted	59
R3TR	TABU	EMCHECK	COFI Check for Add-On Assembly Kit - Disabled one check	Retained	60
R3TR	TABU	EXCEPT2	Object-Specific BC Set Error Case Exception Table	Deleted	55
R3TR	TABU	EXCEPT2T	Exception Justification Text	Deleted	55
R3TR	TABU	FEHT_PROXY2CM PR	Mapping from API and API Method to Business Process	Deleted	58
R3TR	TABU	GEOCD2CLS	Mapping Table: Geocoder ID for ABAP OO Class	Retained	57
R3TR	TABU	GLOSSARY1	Table with Glossary Entries	Deleted	58
R3TR	TABU	GLOSSARY2	Glossary Text Table	Deleted	58
R3TR	TABU	GLOSSARY3	Glossary Table	Deleted	58
R3TR	TABU	IDOC SYN	Syntax Description for Basic Types	Deleted	60
R3TR	TABU	IWREFERENC	IWB: Info Object Where-Used References	Deleted	55
R3TR	TABU	NWS_EXT_PROJN	Data Transport Management Tool Projects	Deleted	58
R3TR	TABU	NWS_EXT_PROJN T	Data Transport Management Tool Project Texts	Deleted	58
R3TR	TABU	OBJSUB	Subobjects	Deleted	55
R3TR	TABU	OBJSUBT	Object: Short Description of Subobject	Deleted	55
R3TR	TABU	POWL_SELCRIT	Criteria Storage	Deleted	57
R3TR	TABU	PPFTFLTVAL	PPF: Value Table for Filter Value of BADI EXEC_METHODCALL	Deleted	57
R3TR	TABU	ROOSFIELD	DataSource Fields	Deleted	60
R3TR	TABU	ROOSOURCE	Table Header for SAP BW OLTP Sources (Relevant from 2.0)	Deleted	60
R3TR	TABU	ROOSOURCET	Texts for an OLTP Source	Deleted	60

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	TABU	RSECACHK	Table for Controlling ABAP Programs	Deleted	54
R3TR	TABU	SALRTCAT	Alert Category: Definition Table	Deleted	60
R3TR	TABU	SALRTCATC	Alert Category: Classifications	Deleted	60
R3TR	TABU	SALRTCATCT	Alert Category: Classification Texts	Deleted	60
R3TR	TABU	SALRTCATT	Alert Category Text Table: Definition Table	Deleted	60
R3TR	TABU	SALRTCCNT	Alerts: Container Table (Customizing)	Deleted	60
R3TR	TABU	SALRTCCNTT	Alerts: Container Text Table	Deleted	60
R3TR	TABU	SAPWLREORG	SAP Workload: Reorganization control	Deleted	55
R3TR	TABU	SDOKLORTAB	SDOK: Tables of Attributes of Relationships for LOIOs	Deleted	57
R3TR	TABU	SDOKLOTAB	SDOK: Tables for logical information objects	Deleted	57
R3TR	TABU	SDOKPHTAB	SDOK: Tables for physical information objects	Deleted	57
R3TR	TABU	SERPT	Reporting: Texts for Tree Structure	Deleted	58
R3TR	TABU	SPERS_OBJ	Personalization object repository	Deleted	53
R3TR	TABU	SPERS_REG	Personalization registration table	Deleted	57
R3TR	TABU	STERM_COMM	Table with Comment Entries for Terms	Deleted	58
R3TR	TABU	STERM_COMP	Additional Components for Term Entry	Deleted	58
R3TR	TABU	STERM_HEAD	Header Table for STERM (SAP-term): Concept Attributes	Deleted	58
R3TR	TABU	STERM_LINK	Link Between Concept and Associated Terms	Deleted	58
R3TR	TABU	STERM_REF	References to Term Entries	Deleted	58
R3TR	TABU	STERM_TEXT	Terminology Text Table	Deleted	58
R3TR	TABU	STXBITMAPS	SAPscript Graphics Management	Deleted	57
R3TR	TABU	SWBRULECOM	WF: Client-specific start conditions	Deleted	57
R3TR	TABU	SWECDOBTYP	Assignment change document/workflow object types	Retained	57
R3TR	TABU	SWP_ADMIN	Customizing Workflow Runtime System	Deleted	60

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	TABU	SXPGCOTABE	Definition of logical operating system commands(SAP table)	Deleted	55
R3TR	TABU	T52B5	Assignment of Values to Objects	Retained	60
R3TR	TABU	T599W	HR Report Categories	Deleted	59
R3TR	TABU	T777D	Info Types - Dialog/Database Assignment	Deleted	60
R3TR	TABU	T778U	Subtypes	Deleted	57
R3TR	TABU	T77ID	Info Types: Enhancements to T777D	Deleted	60
R3TR	TABU	T77PRNL_XML_V	Pension Return: XML Versions	Retained	60
R3TR	TABU	T77S0	System Table	Deleted	60
R3TR	TABU	TABDIRDEVC	Assignment of Development Class to Table Entries	Deleted	57
R3TR	TABU	TACT	Activities that can be protected	Retained	57
R3TR	TABU	TACTT	Activities that can be protected	Retained	58
R3TR	TABU	TAX_APPLI	Applications for Tax Services	Deleted	59
R3TR	TABU	TBANK_PP_APPM	Methods Implemented by Application for Parallel Processing Framework	Deleted	53
R3TR	TABU	TBANK_PP_PPAP PL	Application Types in Parallel Processing Framework	Deleted	53
R3TR	TABU	TBANK_PP_PPAP PLT	Texts on Application Types in Parallel Processing	Deleted	57
R3TR	TABU	TBD51	Attributes of IDoc inbound function modules	Deleted	59
R3TR	TABU	TBD52	Function modules for inbound ALE-EDI	Deleted	60
R3TR	TABU	TBD62	Assignment of change document field to message type	Deleted	59
R3TR	TABU	TBDA2	ALE message active	Retained	59
R3TR	TABU	TBDME	ALE supplement data for EDI message type	Deleted	59
R3TR	TABU	TBE11	BTE Application Indicator	Retained	54
R3TR	TABU	TBE12	Complementary Software Partner	Retained	58
R3TR	TABU	TBE22	Complementary Software Partner Product	Retained	58
R3TR	TABU	TBE22T	Complementary Software Partner Product Description	Retained	58

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	TABU	TBE23	Business Framework: Partner's Active Products	Retained	58
R3TR	TABU	TBE31	Publish&Subscribe BTE: SAP Enhancement	Retained	59
R3TR	TABU	TBE32	Publish&Subscribe BTE: Customer Enhancements	Retained	58
R3TR	TABU	TBRG	Authorization groups	Retained	55
R3TR	TABU	TBRGT	Authorization Group Names	Retained	55
R3TR	TABU	TCMES	Settings for External Scheduling of Order	Retained	57
R3TR	TABU	TCN41	Network Defaults	Retained	57
R3TR	TABU	TDDAT	Maintenance Areas for Tables	Deleted	53
R3TR	TABU	TMENU01	Node Table for General Structure Storage	Deleted	55
R3TR	TABU	TMENU01R	General Structure Storage References	Deleted	55
R3TR	TABU	TMENU01T	General Structure Storage Node Names	Deleted	55
R3TR	TABU	TNODE01	Node Table for General Structure Storage	Deleted	54
R3TR	TABU	TNODE01R	General Structure Storage References	Deleted	54
R3TR	TABU	TNODE01T	General Structure Storage Node Names	Deleted	54
R3TR	TABU	TNODEIMG	Node table for the new IMG	Deleted	53
R3TR	TABU	TNODEIMGR	References for the new IMG	Deleted	53
R3TR	TABU	TNODEIMGT	General Structure Storage Node Names	Deleted	53
R3TR	TABU	TPGP	ABAP/4 Authorization Groups	Deleted	57
R3TR	TABU	TPS31	Process BTE: Alternative Function Modules from SAP	Deleted	54
R3TR	TABU	TRESC	Reserved Names for Customizing Tables/Objects	Deleted	55
R3TR	TABU	TTDTG	SAPscript: Standard Symbols for Word Processing	Deleted	54
R3TR	TABU	TTREE	Definition table for structures	Deleted	60
R3TR	TABU	TTREET	Name of a structure	Deleted	60
R3TR	TABU	TTREE_EXT	Structure enhancement ID	Deleted	60

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	TABU	TTREE_EXTT	Table TTREE_EXT text table	Deleted	60
R3TR	TABU	TTXID	Valid Text IDs	Deleted	57
R3TR	TABU	TTXIT	Texts for Text IDs	Deleted	58
R3TR	TABU	TTXOB	Valid text objects	Deleted	57
R3TR	TABU	TTXOT	Short Texts for Text Objects	Deleted	58
R3TR	TABU	TTYP	Object Types for Accounting	Retained	57
R3TR	TABU	TVDIR	View Directory	Deleted	53
R3TR	TABU	TVIMF	User routines called from view maintenance	Deleted	54
R3TR	TABU	UCUW001	Personalization Workbench settings	Deleted	53
R3TR	TABU	UCUW011	Personalization Workbench settings	Deleted	53
R3TR	TABU	UJO_FACTORY	BPC: System table for BPC Factories	Deleted	58
R3TR	TABU	UJO_PARAM_CHECK	BPC: Check IMG parameters	Deleted	58
R3TR	TABU	UJA_APPL_COMP	Observer list for application components	Deleted	58
R3TR	TABU	UJA_DIM_OBSERVER	BPC: Dimension observer	Deleted	58
R3TR	TABU	UJA_HIER_CLASS	BPC: Hierarchy class	Deleted	58
R3TR	TABU	UJO2_ENGINE_LIST	Registered Query Engines and their priorities	Deleted	58
R3TR	TABU	UJT_TRANS_POST	BPC: Post action after transport	Deleted	58
R3TR	TABU	UJT_VAL_CONFIG	BPC: Transport setting for configuration	Deleted	58
R3TR	TABU	URL_EXITS	Definition of workplace node types	Deleted	55
R3TR	TABU	USR10	User master authorization profiles	Deleted	55
R3TR	TABU	USR11	User Master Texts for Profiles (USR10)	Deleted	55
R3TR	TABU	USR12	User Master Authorization Values	Deleted	54
R3TR	TABU	USR13	Short Texts for Authorizations	Deleted	54
R3TR	TABU	UST10C	User master: Composite profiles	Deleted	55
R3TR	TABU	UST10S	User master: Single profiles	Deleted	55
R3TR	TABU	UST12	User master: Authorizations	Deleted	54

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	TABU	WWWDATA	INDX-Like Table for Repository of WWW Objects	Deleted	60
R3TR	TABU	WWWPARAMS	Parameter settings for Web Reporting	Deleted	60
R3TR	TDAT	/BA1/P3_EC	Modular Costing	Deleted	57
R3TR	TDAT	ADDRESS	Customizing addresses	Retained	57
R3TR	TDAT	ADDRESS_4.6	Customizing Addresses Release > 4.5	Retained	57
R3TR	TDAT	BDFG	Generate ALE Interfaces from Function Module	Deleted	59
R3TR	TDAT	EDISEGMENT	EDI: Table entries for segm.man- agmt/modif.concept	Deleted	59
R3TR	TDAT	LT_VARIANT	ABAB/4 List viewer report display variants	Deleted	55
R3TR	TDAT	SUSPR	Authorization profiles	Deleted	54
R3TR	TDAT	WCF_GENIL_SAP	Transport object for Genil - SAP system	Deleted	60
R3TR	TDAT	WDY_CONF_USER	Configuration Data / Personalization Data with Description	Deleted	54
R3TR	TEXT	*	SAPscript Text	Deleted	53
R3TR	TOBJ	*	Definition of a Maintenance and Transport Object	Deleted	53
R3TR	TRAN	*	Transaction	Deleted	53
R3TR	TTYP	*	Table Type	Deleted	53
R3TR	TYPE	*	Type Group	Deleted	53
R3TR	UDMO	*	Data Model	Deleted	58
R3TR	UENO	*	Entity Type	Deleted	58
R3TR	VCLS	*	View Cluster	Deleted	53
R3TR	VDAT	/BA1/ PO_MV_STRAT	View Maintenance for Strategies	Deleted	57
R3TR	VDAT	/BA1/P3_MV_EC	Field Catalog for Calculation Tool	Deleted	57
R3TR	VDAT	/BA1/ P4_MV_APPL	Maintenance View for Application Indicator	Deleted	57
R3TR	VDAT	/BA1/P4_MV_SET	Maintenance View for Derivation Settings	Deleted	57
R3TR	VDAT	/IWFND/ V_MGDEAM	Assign SAP System Aliases to OData Service	Retained	57

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	V DAT	/PRAPG/ V_XML_IN	Mapping of XML Tags to Indicators	Retained	60
R3TR	V DAT	/PRAPG/ V_XML_VER	Pension Return XML Versions Maint. View	Retained	60
R3TR	V DAT	/SAPPO/ SSAPCMPNT	Software Component of Postprocessing Office	Deleted	54
R3TR	V DAT	/SAPPO/ VA_BPROC	Activate Creation of Postprocessing Orders	Deleted	58
R3TR	V DAT	/SAPPO/ VSSCRN_AS	Tab Page Display in Object Area	Deleted	58
R3TR	V DAT	/SAPPO/VSVALI-DOT	Define Permitted Object Types	Deleted	58
R3TR	V DAT	/SAPPO/ VS_BPROC	Business Processes (System) of Postprocessing Office	Deleted	54
R3TR	V DAT	/SAPPO/ VS_CMPNT	Software Components (System) of Postprocessing Office	Deleted	55
R3TR	V DAT	/SAPPO/ VS_OBJBOR	Methods for each Component of Postprocessing Office	Deleted	54
R3TR	V DAT	/SAPPO/ VS_OBJECT	Object Types of Postprocessing Office	Deleted	54
R3TR	V DAT	/UI2/V_SEMOBJ	Semantic Objects - SAP Delivery	Deleted	58
R3TR	V DAT	ARBFND_V_APPL	Control tables of the Ariba integration layer	Deleted	54
R3TR	V DAT	ARBFND_V_OBJT	Control tables of the Ariba integration layer	Deleted	54
R3TR	V DAT	COMV_PARTNER_FCT	Partner Functions	Retained	58
R3TR	V DAT	CRMV_ACT_CAT_ASS	Assign Activity Categories to Transaction Types	Retained	58
R3TR	V DAT	CRMV_EVC_ALL	Maintenance of Callback Functions	Deleted	58
R3TR	V DAT	CRMV_FUNC_ASSIGN	Assignment: Event Handler Modules for Object Function	Deleted	58
R3TR	V DAT	CRMV_GIL_COMP	Maintenance View for Table CRMC_GIL_COMP	Deleted	60
R3TR	V DAT	CRMV_OBJECT_FUNC	Event Handler: Assignment Object - Functions	Deleted	58
R3TR	V DAT	CRMV_OBJ_FUNC	Event Handler: Object Functions	Deleted	58

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	VDAT	CTSRESNAME	Checked with BC-CTS-ORG - Jens Willibald - Partner request see Incident 0020079747 0000022572 2016	Deleted	61
R3TR	VDAT	DCFLV_BO	Decoupling: Business Object (Entity)	Deleted	59
R3TR	VDAT	DCFLV_BO_OPERAT	Decoupling: Business Object-Specific Structures	Deleted	59
R3TR	VDAT	DCFLV_BO_STEP	Decoupling: Implementation of Steps per Business Object	Deleted	59
R3TR	VDAT	DCFLV_STEP	Decoupling: Step	Deleted	59
R3TR	VDAT	DCFLV_STEP_IF	Decoupling: Interfaces for Steps	Deleted	59
R3TR	VDAT	DCFLV_STEP_VAR	Decoupling: Step variants with steps	Deleted	59
R3TR	VDAT	DCFLV_VARIANT	Decoupling: Step Variants (Entity)	Deleted	59
R3TR	VDAT	ECHV_PROCESS	Process Data for Error and Conflict Handler	Deleted	54
R3TR	VDAT	FEHV_PROXY2CM PR	Mapping from API and API Method to Business Process	Deleted	54
R3TR	VDAT	FPM_V_ADAPT_DIM	FPM Adaptation Schema Dimension	Deleted	60
R3TR	VDAT	FPM_V_ADAPT_SCHM	FPM Adaptation Schema	Deleted	60
R3TR	VDAT	POWL_V_QUERY	View: Query definition	Deleted	57
R3TR	VDAT	POWL_V_QUERY_R	View: Query - Role assignment	Deleted	57
R3TR	VDAT	POWL_V_TYPE	View: Type definition	Deleted	57
R3TR	VDAT	POWL_V_TYPE_R	View: Type - Role assignment	Deleted	57
R3TR	VDAT	SWFDVEVTY2	Event Type Linkages	Deleted	57
R3TR	VDAT	VED2_TEDE2	Process codes, inbound	Deleted	60
R3TR	VDAT	VEDIEDIFCT	IDoc: Assignment of FM to Log. Message and IDoc Type	Deleted	59
R3TR	VDAT	VEDI_EDMSG	EDI: Logical Message Types	Deleted	57
R3TR	VDAT	V_ARC_USR	Customizing View for Archiving	Deleted	57
R3TR	VDAT	V_BALSUB	Application log: Subobject maintenance view	Retained	53
R3TR	VDAT	V_BRG_54	Authorization groups for tables and views	Retained	54

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	VDAT	V_CNVMBTACTIVITT	MBT PCL Activity text maintenance	Deleted	60
R3TR	VDAT	V_CNVMBTACTIVITY	MBT PCL Activity maintenance	Deleted	60
R3TR	VDAT	V_CNVMBTACTPAR	MBT PCL Activity parameters	Deleted	60
R3TR	VDAT	V_CNVMBTACTTSREF	Reference activities to troubleshooting processes	Deleted	60
R3TR	VDAT	V_CNVMBTCOPYVAR	MBT PCL Copy Variants for Proc Type switch	Deleted	60
R3TR	VDAT	V_CNVMBTPACK	View for package and description	Deleted	60
R3TR	VDAT	V_CNVMBTPROC-TYPE	MBT PCL : Definition of process types	Deleted	60
R3TR	VDAT	V_CNVMBTRESE-TACT	Activities to be reset	Deleted	60
R3TR	VDAT	V_CNVMBTRE-SETVAR	Activity reset variant	Deleted	60
R3TR	VDAT	V_CNVMBTSPEZL	MBT PCL SPEZTAB entries for package load procedure	Deleted	60
R3TR	VDAT	V_CNVMBTSTATE	Maintenance view for CNVMBTSTATE	Deleted	60
R3TR	VDAT	V_CNVMBTTSDEF	Troubleshooting process definition	Deleted	60
R3TR	VDAT	V_CNVMBTTSH	Troubleshooting process header	Deleted	60
R3TR	VDAT	V_CNV_OBJSUB	Subobjects	Deleted	55
R3TR	VDAT	V_DDAT_54	Authorization group assignment to table/view	Deleted	57
R3TR	VDAT	V_EDIDOCMAP	Mapping of IDoc Types	Deleted	60
R3TR	VDAT	V_EDISEGMAP	Mapping of IDoc Segments	Deleted	60
R3TR	VDAT	V_FPB_PERSAPPL	Maintenance View of the Personalization Hierarchy	Deleted	57
R3TR	VDAT	V_HRSFI_R_PARA_S	Parameters	Deleted	60
R3TR	VDAT	V_SDOKSTCA	SDOK: Maintain Categories	Deleted	57
R3TR	VDAT	V_T001K_ASSIGN	Assignment Plant - Company Code	Retained	57
R3TR	VDAT	V_T001W	Plants	Retained	57
R3TR	VDAT	V_T003O_PS	Maintain Network Types	Retained	57
R3TR	VDAT	V_T009B	Period assignment	Deleted	55

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	VDAT	V_T024A	Maintenance planner group	Retained	57
R3TR	VDAT	V_T024D	MRP Controllers	Retained	57
R3TR	VDAT	V_T399X_NN	Network type parameters: Overview	Retained	57
R3TR	VDAT	V_T430_CN	Activity Control Key	Retained	57
R3TR	VDAT	V_T50BA	B2A: Valid Transfer Formats and Classes	Retained	60
R3TR	VDAT	V_T50BB	B2A: Standard Transfer Format	Retained	60
R3TR	VDAT	V_T50BD	HR-B2A: Valid Status	Deleted	60
R3TR	VDAT	V_T50BF	HR-B2A: Maintain Method Table for Statuses/Substatuses	Deleted	60
R3TR	VDAT	V_T50BG	HR-B2A: Valid Manual Status Change	Deleted	60
R3TR	VDAT	V_T50BK	HR-B2A: Constants	Retained	60
R3TR	VDAT	V_T52B4	Valid Attribute Values	Deleted	58
R3TR	VDAT	V_T5961	Areas	Retained	60
R3TR	VDAT	V_T77PRNL_XML_IN	Pension Return: XML Tags Mapping Indicator	Retained	60
R3TR	VDAT	V_T77PRNL_XML_V	Pension Return: XML Versions	Retained	60
R3TR	VDAT	V_T7XSSCE_GRP	Group definition settings for ESS - Table CLASS G cannot be deleted	Retained	60
R3TR	VDAT	V_TBCCA	Bar Chart: Chart	Deleted	57
R3TR	VDAT	V_TBCC	Bar Chart: Color Assignment	Deleted	57
R3TR	VDAT	V_TBCCF	Bar Chart: Form Assignment	Deleted	57
R3TR	VDAT	V_TBCCG	Bar chart: Graphic profile	Deleted	57
R3TR	VDAT	V_TBCL	Bar chart: Graphic element	Deleted	57
R3TR	VDAT	V_TBCCO	Bar chart: Option profile	Deleted	57
R3TR	VDAT	V_TBCT	Bar chart: Field definition	Deleted	55
R3TR	VDAT	V_TBD51	Characteristics of Inbound Function Modules	Deleted	59
R3TR	VDAT	V_TBD62	Change document items for message type	Deleted	59
R3TR	VDAT	V_TBDA2	Activate Change Pointers for Message Type	Retained	59
R3TR	VDAT	V_TBDME	Additional Data for Message Type	Deleted	59

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	VDAT	V_TBFM	Bar Chart: Form Definition	Deleted	57
R3TR	VDAT	V_TBHL	Bar Chart: Color Definition	Deleted	57
R3TR	VDAT	V_TCNC	Network/Hierarchy: Frames	Deleted	55
R3TR	VDAT	V_TCNG	Network/Hierarchy: Graphics Profile	Deleted	55
R3TR	VDAT	V_TCNL	Network/Hierarchy: Links	Deleted	55
R3TR	VDAT	V_TCNN	Network/Hierarchy: Nodes	Deleted	55
R3TR	VDAT	V_TCNO	Network/Hierarchy: Option Profile	Deleted	55
R3TR	VDAT	V_TFAT	Network/Hierarchy: Field Definition	Deleted	55
R3TR	VDAT	V_TFMT	Network/Hierarchy: Form Definition	Deleted	55
R3TR	VDAT	V_TGMF	VarChart Graphics: Files for including symbols	Deleted	57
R3TR	VDAT	V_THLT	Network/Hierarchy: Color Definition	Deleted	55
R3TR	VDAT	V_TPGP	Maintain tables TPGP and TPGPT	Deleted	57
R3TR	VDAT	V_TRESN	Reserved namespaces	Deleted	61
R3TR	VDAT	V_TTXIDI	Text IDs allowed	Deleted	57
R3TR	VDAT	V_TTXOBI	Valid Text Objects	Deleted	57
R3TR	VDAT	V_TTZEX	Convert time zone ID ext. -> SAP	Retained	58
R3TR	VDAT	V_TVIMF	FORM routines to be called from view maintenance	Deleted	54
R3TR	VDAT	V_TWPC_AR-RAYTP	Definition of Column Groups	Retained	58
R3TR	VDAT	V_TWPC_COL_ERP	Column Definition	Retained	58
R3TR	VDAT	V_TWPC_DATAVW	Data View	Retained	58
R3TR	VERS	*	Information about component version	Retained	53
R3TR	VIEW	*	View	Deleted	53
R3TR	W3HT	*	Web Reporting/Internet Transaction Server HTML Templates	Deleted	53
R3TR	W3MI	*	Web Reporting/Internet Transaction Server MIME Types(binary	Deleted	53
R3TR	WAPA	*	BSP (Business Server Pages) Application	Deleted	53

Program ID	Object type	Object name	Description	Handling	Supported as of SPAM/ SAINT Version
R3TR	WDCA	*	Web Dynpro Application Configuration	Deleted	53
R3TR	WDCC	*	Web Dynpro Component Configuration	Deleted	53
R3TR	WDCP	*	Web Dynpro CHIP	Deleted	60
R3TR	WDYA	*	Web Dynpro Application	Deleted	55
R3TR	WDYN	*	Web Dynpro Component	Deleted	55
R3TR	WEBI	*	Virtual End Point	Deleted	54
R3TR	XINX	*	Extension Index	Deleted	57
R3TR	XPRA	*	Program Run After Transport	Retained	55
R3TR	XSLT	*	Transformation	Deleted	54

13 Additional Information

This document contains information about the following topics:

- [Overview: Import Tools \[page 115\]](#)
- [Modifications and Their Consequences \[page 115\]](#)
- [Conflicts \[page 122\]](#)
- [Examples: Attributes in Software Delivery Assembler \[page 128\]](#)
- [CDs for Add-On Deliveries \[page 130\]](#)
- [Troubleshooting \[page 133\]](#)
- [Paths to Documentation in SAP Help Portal \[page 133\]](#)
- [Terminology \[page 137\]](#)

13.1 Compatibility of SAP NetWeaver Releases



The concept of downward-compatible SAP NetWeaver releases was introduced alongside SAP NetWeaver enhancement packages. Downward compatibility means that ABAP software developed for a specific SAP NetWeaver release can also run on higher downward-compatible SAP NetWeaver releases. This means that a package given attributes for a specific SAP NetWeaver release (either a support package, installation package, or upgrade package) can be installed in higher downward-compatible SAP NetWeaver releases.


The compatibility rules are shipped using the ABAP software logistics tools Support Package Manager (transaction SPAM) and SAP Add-On Installation Tool (transaction SAINT).

➔ Recommendation

Always import the newest version of a SPAM/SAINT update before working with these tools.

















































































































































There are three downward compatibility statuses:

-  not compatible:
Packages in this SAP NetWeaver release cannot be installed in the other SAP NetWeaver release. It is not possible to give packages attributes and deliver them in such a way that they can be imported into both SAP NetWeaver releases.
-  compatible with restrictions:
In standard cases, packages in this SAP NetWeaver release cannot be installed in the other SAP NetWeaver release. If it is possible to establish that the ABAP software can run in both SAP NetWeaver releases, it is possible to give the packages attributes enabling them to be imported into both SAP NetWeaver releases. The import behavior for all target SAP NetWeaver releases must be checked carefully before a package is shipped to customers. It is possible that the import can fail, for example if referenced objects do not exist in the target system. The import can only be enabled by defining an alternative import condition for the target SAP NetWeaver release. Here, the original SAP NetWeaver release must be removed from the alternative import condition and the higher downward-compatible SAP NetWeaver release entered instead.

-  fully compatible:
 A package built for a specified SAP NetWeaver release can be installed in the other SAP NetWeaver release without any further actions.

The following matrix contains the current compatibility rules. The columns indicates the SAP NetWeaver release where the packages is installed and the rows indicate the original SAP NetWeaver release of the package.

Table 21: Compatibility of SAP NetWeaver Releases

SAP_BA-SIS	6.20	6.40	7.0	7.0 EHP1	7.00 EHP2	7.1	7.1 EHP1	7.2	7.3	7.3 EHP1	7.4	7.5
6.20												
6.40												
7.00												
7.0 EHP1												
7.00 EHP2												
7.1												
7.1 EHP1												
7.2												
7.3												
7.3 EHP1												
7.4												
7.5												

Example

- A package was built for SAP NetWeaver 7.0 EHP1. This package can be imported into the SAP NetWeaver Releases 7.0 EHP1, 7.0 EHP2, 7.3 EHP1, 7.4, and 7.5.
- A package was built for SAP NetWeaver 7.1. This package can be imported into the SAP NetWeaver Releases 7.1 and 7.1 EHP1. If the appropriate attributes are set, it is also possible to import the package into the SAP NetWeaver releases 7.2, 7.3, 7.3 EHP1, 7.4, and 7.5.
- Your system has the release level SAP NetWeaver 7.3 EHP1. Packages for SAP NW Releases 7.0, 7.0 EHP1, 7.0 EHP2, and 7.3 EHP1 can be imported in standard cases. In the case of packages built in the SAP NW releases 7.1, 7.1 EHP1, 7.2, and 7.3, the alternative import conditions must be checked for importability.

13.2 Overview: Import Tools


The import tools for add-on packages are:

Table 22:

Tool	Description
SAP Add-On Installation Tool (transaction SAINT)	Installs and upgrades add-on packages directly from the SAP system.
Support Package Manager (transaction SPAM)	Imports support packages (such as add-on support packages or conflict resolution transports)
Software Update Manager (SUM)	Controls SAP system upgrades (based on ABAP) To update add-ons during SAP system upgrades, add-on upgrade packages and add-on exchange packages can be included in the upgrades.

Note

Detailed information about the tool in question can be found in SAP Add-On Installation Tool and Support Package Manager

SAP system upgrades using SUM are described in *Update of SAP Systems Using Software Update Manager* for the SL toolset release in question (linked under <https://help.sap.com/sltoolset>) and individual upgrade guides are available in SAP Service Marketplace <https://service.sap.com/instguides> .

13.3 Modifications and Their Consequences

You must check whether your add-on requires modifications to be made to the standard SAP system before you start development of the add-on.

It is technically possible to make modifications to application components in the layer *R* for any add-ons creating using SAP Add-On Assembly Kit. **We strongly advise against creating modifying add-ons**, however, since modifications change the behavior of the entire system. Modifying add-ons are not certified by SAP. If you want to make modifications to an object, you must verify that no other SAP components could still use it. This means that modifications entail significantly more work and also have consequences in the following areas:

- Add-on development landscape and maintenance landscape
If you want to make modifications, this has consequences for both the add-on development landscape and the maintenance landscape.
If you develop non-modifying add-ons, you can develop multiple add-ons in a single system (if certain prerequisites are met). This is not the case for modifying add-ons. You can develop only one add-on in each system landscape.
For more information, see [Developing Multiple Add-Ons in a System Landscape \[page 35\]](#).
- Imports

When your customers use your add-on, it is not verifiable that they can import SAP support packages into the add-on system, since these packages could overwrite the add-on modifications. It is usually necessary to create CRTs.

In addition, you cannot ensure that customers are able to import your add-on. If a different add-on modifies the same objects and this add-on is already installed in the system, it is not possible to install both at the same time. Similarly, it may no longer be possible to import another add-on if your add-on is already installed and both add-ons modify the same object. Also read the section about conflicts with add-on components in the same layer under [Conflicts in Installations/Upgrades of an Add-On \[page 123\]](#).

- **Maintenance**

Any modifications made to standard SAP development objects increase the way add-on development objects are merged with the standard SAP system. This makes it impossible to perform maintenance tasks and upgrades without dependencies.

You must compare any changes imported into the add-on development system in support packages with the add-on modifications and synchronize (adjust) them if necessary. You must create conflict resolution transports (CRTs). This applies to every support package created by SAP for the component modified by your add-on and that contains conflicts with your modifications. This is an action that is prone to errors. Moreover, this makes your add-on dependent on the SAP maintenance cycle, since you need to repeat this action until the end of the maintenance cycle of the SAP release in question.

More Information

- In many cases, you can avoid making modifications by using enhancement techniques instead, such as enhancement spots or appends. If your development work is based on SAP NetWeaver 7.0, you can use Enhancement Framework. Check the viability of these options thoroughly before resorting to modifications. See also: [Additional Information About Enhancements and Modifications \[page 48\]](#).
- If modifications are essential regardless, you must still follow a number of [rules \[page 119\]](#).
- Also read the information under [Conflicts \[page 122\]](#) and the associated sections.

13.3.1 Setting Up a Development Landscape for Modifying Add-On

If you want your add-on to support multiple SAP releases, you require a separate development landscape for each release in question. To set up a development landscape for add-ons with modifications, proceed as follows:

Procedure

1. Create the new development system as a copy of the first consolidation system.
2. Choose one of the following options in for the client layout:
 - In the new development system, use the former test client as the new development client and the former delivery client as the new customizing client.
 - If you want to base your development work on the delivered customizing of the predecessor release, you can also create the new development client as a copy of the new customizing client.

3. After making the copy, perform an SAP system upgrade to the new SAP release in the development system. If required, adjust the modified objects (using transactions SPPD and SPAU). You must also update SAP Add-On Assembly Kit during the upgrade. Then check the [settings \[page 39\]](#) in the development system. You must, for example, update the name of the add-on release.
4. Create the new consolidation system as an installation of a new SAP system or as a copy of an installed SAP system. [Install SAP Add-On Assembly Kit \[page 38\]](#) in the new consolidation system and perform the required [settings \[page 39\]](#).
5. If you want to deliver your add-on in multiple languages, set up the translation environment in the new consolidation system in the same way as in the existing consolidation system. For more information, see the documentation [Setting Up and Coordinating Translation](#) in SAP Help Portal.
6. In the customizing client of the new development system, create a transport request by including the component piece list of the predecessor release. Choose [Transport of copies](#) as the request type.
7. Export the transport of copies in the translated languages of the first development landscape. When doing this, make sure that the parameter LSM is set to ALL and that the parameter LANGUAGE contains all translated languages from the first development landscape. The value MASTER is set for the parameter LSM by default (see also [Setting the Parameters LANGUAGE and LSM \[page 43\]](#)).
8. Import the transport of copies into the delivery client of the new consolidation system.
9. Register the transport of copies as a component piece list of the predecessor release. This piece list is required later when the exchange component piece list is created. To do this, start Software Delivery Composer in the delivery client of the new consolidation system. In the initial screen, choose ► [Delivery for Delivery Request](#) ► [Register](#) .
10. Instruct your developers to verify in the new development system which of the add-on objects need to be modified or deleted in the new SAP release. The add-on objects are in the component piece list of the predecessor release. Once all adjustments have been made, transport all add-on objects (including the deleted objects) to the new consolidation system in a consolidation transport. This transport becomes the basis of your new supported SAP release.

Example

The modifying add-on EFGH, Release 100, was developed on SAP NetWeaver 7.0 (EFGH 100_700). The add-on is now developed further as the new Release 200 on SAP NetWeaver 7.4 (EFGH 200_740). The figure illustrates the setup of the new development landscape.

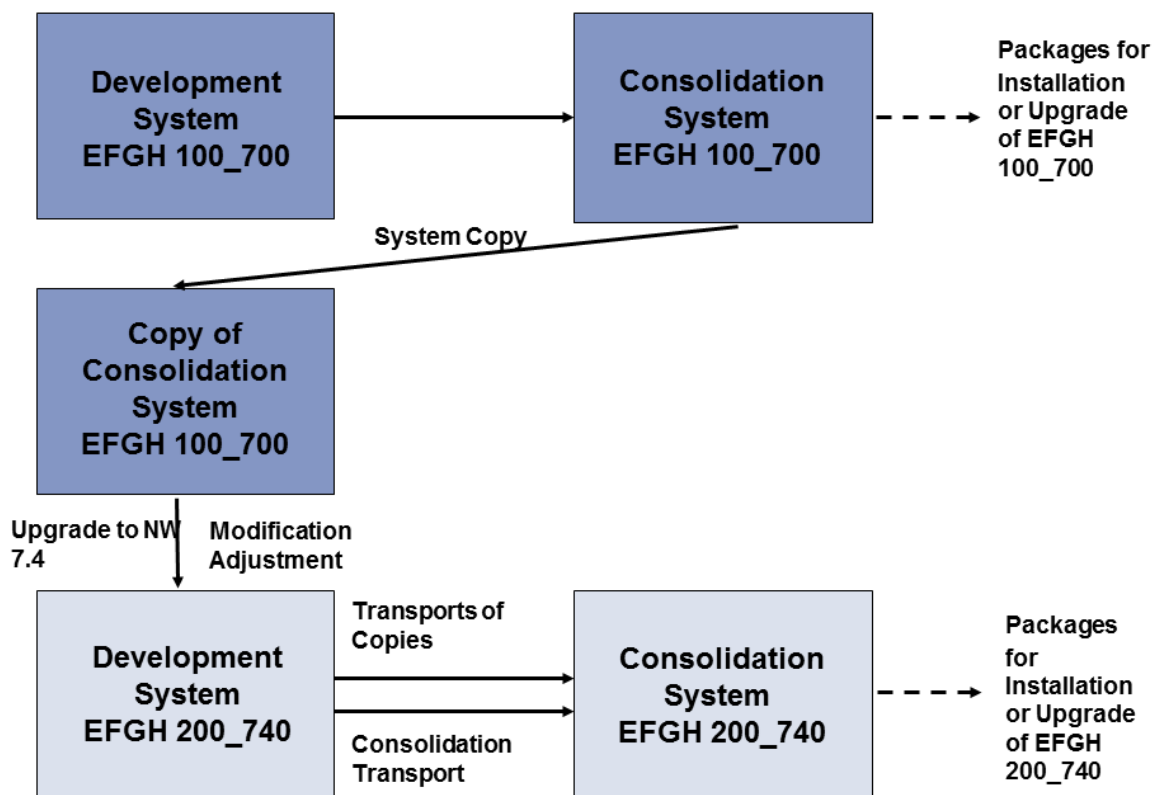


Figure 5: Setup of the New Development Landscape for the Modifying Add-On EFGH 200_740

➔ Recommendation

You can also apply this procedure to non-modifying add-ons. This is not recommended, however, since upgrades in this case require more work.

i Note

In all cases, you must reconstruct the state you require for your [delivery strategy \[page 19\]](#), in particular the release state and support package state, when you are setting up the SAP system.

When you are setting up the system landscape, also take care when creating the [client layout \[page 36\]](#).

13.3.2 Maintenance for a Modifying Add-On

If your add-on modifies the standard SAP system, you must perform a modification adjustment for SAP support packages, if these packages overwrite your modifications.

In the case of modifying add-ons, you can perform the modification adjustment in the maintenance system to create CRTs. The two-system landscape also enables objects without versioning to be compared. The second

system functions as a buffers, since it is impossible to predict how and when SAP support packages affect your add-on, and how much work is required to create CRTs.

In the case of system landscapes with modifications, the modification adjustment is an additional maintenance task to those tasks described in section [System Landscape for Add-On Maintenance \[page 32\]](#).

13.3.3 Rules for Developing a Modifying Add-On

If it becomes necessary to make modifications to standard SAP objects, we recommend the "less is more" principle. Note also the following rules:

- Modifications to objects in the software components SAP_BASIS and SAP_ABA plus modifications to other add-ons are not permitted.
Modifications to ABAP Dictionary objects from SAP_BASIS can be lost in upgrades. After an upgrade, these objects exist in their original form again. The system does not support modification adjustments to objects from SAP_BASIS.
- Any modifications made by your add-on can only be done in an SAP main component or application component (such as SAP_HR or SAP_APPL).
- Use the Modification Assistant. For more information, see SAP Help Portal under [The Modification Assistant](#)
- Also note the further restrictions in this table:

Table 23:

Modification	Development (Add-On Installation Package, Add-On Upgrade Package, Add-On Exchange Package)	Maintenance (Add-On Support Package, Conflict Resolution Transport)
New modification objects	Not recommended	Not allowed
New function modules in SAP function groups	Not allowed	Not allowed
Changes to the standard SAP user interface	Not allowed	Not allowed
Dictionary changes that modify database tables (such as adding fields or includes)	Not allowed	Not allowed
Modification to non-versionable objects in other software components (such as table entries in SAP tables or logical transport objects); it is virtually impossible to perform an adjustment for these modifications using conflict resolution transports	Not allowed	Not allowed
Delete or rename standard SAP objects	Not allowed	Not allowed

Modification	Development (Add-On Installation Package, Add-On Upgrade Package, Add-On Exchange Package)	Maintenance (Add-On Support Package, Conflict Resolution Transport)
Modifications to standard SAP customizing	Not allowed	Not allowed
Modifications to domains	Not allowed	Not allowed
Customer exits	Not allowed	Not allowed
Use of customizing includes and system includes	Not allowed	Not allowed

Note

Check your modifications at regular intervals in the Modification Browser (transaction SE95).

13.3.4 Creating Conflict Resolution Transports

Use

Note

Conflict resolution transports (CRTs) are not required for non-modifying development projects.

CRTs adapt an add-on to SAP support packages. Before this can happen, you must first detect the conflicts between the SAP support packages and all add-on deliveries for an add-on release.

Note

In the case of CRTs, the term SAP support packages refers to any support packages for the main component or application component where the add-on made modifications.

For add-ons based on SAP Web AS 6.20 or higher, Software Delivery Composer supports conflict detection and the creation of CRTs for support packages in layer R (application components) and one add-on from the layer C. (See also: [Software Component Layers \[page 14\]](#))

Note also the restrictions that apply to permitted changes. You can find these in [Rules for Add-On Maintenance \[page 76\]](#).

Procedure

First perform the steps under Actions in the Maintenance System and then the actions required for corrections in the consolidation system.

Actions in the Maintenance System

1. Load the support packages in question from SAP Support Portal under <https://support.sap.com/swdc>.
2. Use Support Package Manager to import the support packages into the maintenance system.
3. Use transactions SPDD and SPAU to adjust the modifications in Support Package Manager.
 1. Adjust the direct conflicts.
 2. Adjust any dependent objects.
 3. Adjust any copied objects.
 4. Choose *Reset to Original* for any preliminary corrections in the SAP support package.
4. If required, you can also make corrections to the add-on software.
5. Verify that all objects have been processed in transaction SPAU. Then release the following requests:
 1. All modification adjustment requests
 2. Any transport requests containing corrections to add-on software

Actions for Corrections in the Consolidation System

1. Use Support Package Manager to load the SAP support packages imported into the maintenance system into the consolidation system for corrections (consolidation system for short).
2. Use Support Package Manager to import these support packages into the consolidation system and execute transaction SPDD (adjust dictionary objects) again if required.
3. Import the transport requests mentioned in the last step into the consolidation system.

If imports have errors or if corrections are missing, make the appropriate changes in the maintenance system and transport them into the consolidation system.
4. Detect the conflicts between the imported SAP support packages and all add-on deliveries for an add-on release.
 - To do this, compare the current add-on scope for the add-on release in question (consisting of add-on installation package, add-on upgrade packages, add-on support packages, and shipped conflict resolution transports) with the object lists of the SAP support packages you have just imported.
 - The intersection of the add-on scope in question and the SAP support packages contains all modified objects that were overwritten and the overwritten objects that you previously reset to original.
 - You must deliver the objects that were reset to original and that now reoccur in the SAP support packages to your customers again. This adapts the customer system in question to the new add-on correction level.
 - Use Software Delivery Composer to detect the conflicts. On the initial screen of Software Delivery Composer, choose ► *Utilities* ► *Display/Determine Conflicts* . If this check does not detect any conflicts, you do not need to create a CRT.

Software Delivery Composer checks for conflicts in the following objects:

 - Repository objects
 - Logical transport objects

Software Delivery Composer does not check for conflicts between table entries.
5. If you need to create a CRT, create the delivery using the AAK tools (see [Creating a Delivery \[page 57\]](#)). Create the change list of the CRT by including the transport requests imported into the consolidation system:
 1. Transport requests with modification adjustment

2. Any transport requests containing corrections to add-on software

When you use Software Delivery Composer to create deliveries with type CRT, you must select the main component or application component where you detected conflicts. Software Delivery Composer then adds the objects that cause the conflicts to the CRT change list automatically.

Once you have added all requests that are relevant for the delivery to the change list, release the CRT.

If you are creating CRTs in parallel for your add-on for multiple releases, you must use the extended attribute EQUIVALENT when you register the CRT in Software Delivery Assembler. This attribute associates support packages or CRTs with different add-on releases. It is intended to preserve corrections from the CRTs in the higher release in SAP system upgrades. For more information about the attribute, EQUIVALENT, see the online documentation in Software Delivery Assembler under [Tab: Extended Attributes](#).

13.4 Conflicts

If your add-on contains modifications, that is, it delivers changes to a different software component to your own, you may encounter conflicts when installing it.

Conflicts can occur in the following cases:

- During installation: When installing/upgrading an add-on with Add-On Installation Tool
- During maintenance: When importing Support Packages with Support Package Manager

When packages are imported, Add-On Installation Tool and Support Package Manager check for conflicts with existing add-ons or Support Packages. Conflicts only occur with modifying add-ons. If, for example, an add-on modifies an object that is then delivered again with a SAP Support Package, the person who created the add-on has to resolve this conflict. Otherwise, the Support Package cannot be imported. If you have created an add-on that contains modifications, you have to react to any conflicts in order to ensure that your add-on can still run. Conflict resolution causes a lot of extra effort, and is sometimes not possible.

For further information, see:

[Background Information: Conflict Check \[page 122\]](#)

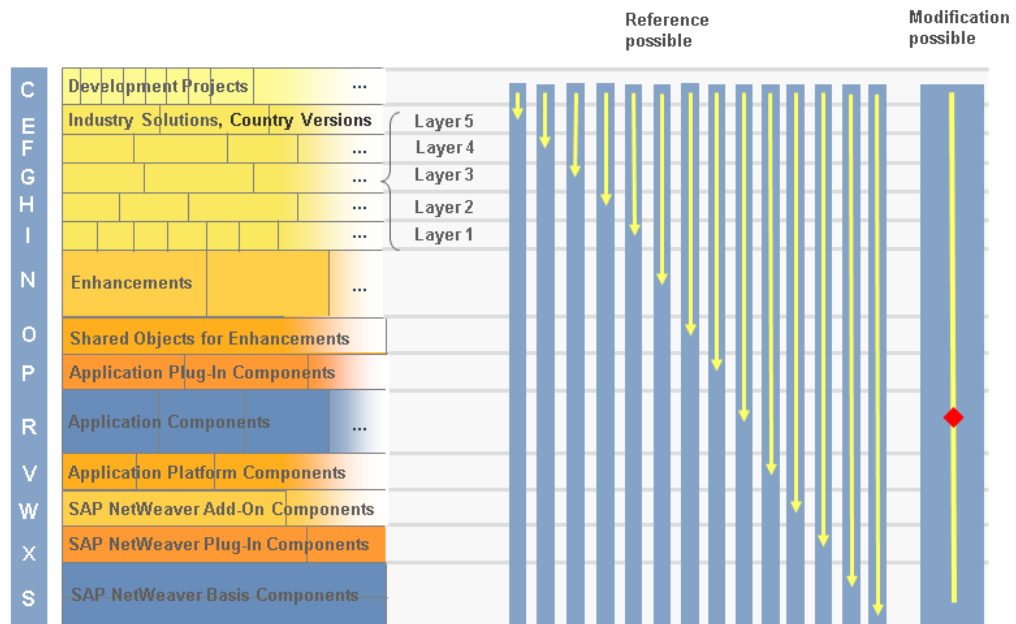
[Conflicts When Installing/Upgrading an Add-On \[page 123\]](#)

[Conflicts When Importing Support Packages \[page 127\]](#) :

13.4.1 Background Information: Conflict Check

From the introduction of the software component hierarchy in SAP Web AS 6.20, add-ons created using SAP Add-On Assembly Kit are in level 'C'. Add-ons in level 'C' can reference objects from components in lower levels. Modifications, however, can only be made to objects from software components in layer 'R'. SAP Add-On Assembly Kit supports conflict checks and makes it possible to create conflict resolution transports only for conflicts of the add-on with support packages from layer 'R'.

References and Modifications in the Software Component Hierarchy



13.4.2 Conflicts When Installing/Upgrading an Add-On

When an add-on is installed or upgraded, Add-On Installation Tool checks for the following conflicts:

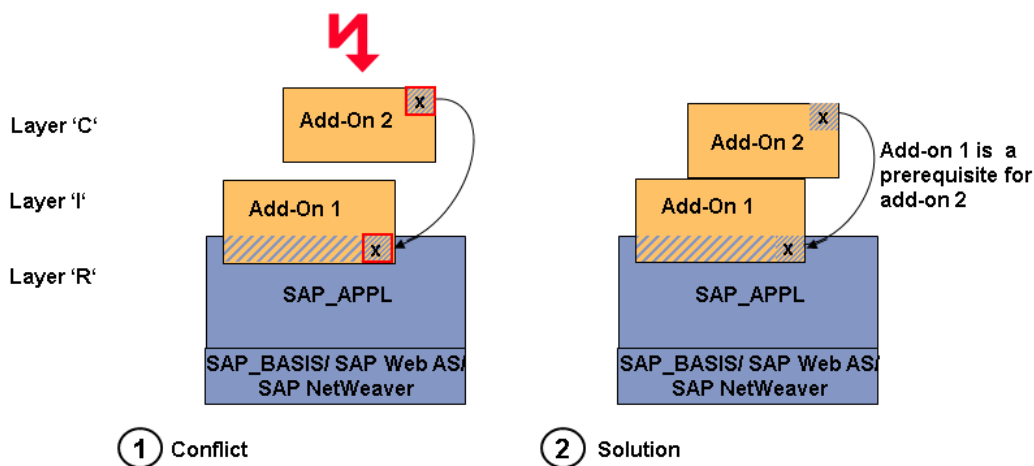
Conflicts with Add-On Components in Lower Layers

Add-On Installation Tool checks for conflicts with add-on components in lower layers.

Example: Add-on 1 in layer 'I' and add-on 2 in layer 'C' modify the same object in component SAP_APPL (in layer 'R'). Add-on 1 has already been installed. During installation of add-on 2, Add-On Installation Tool detects a conflict and aborts the installation process.

If you want add-on 1 and add-on 2 to be installed in one system at the same time, and you created add-on 2 yourself, you now need to resolve the conflict. To do this, you need to define add-on 1 as a prerequisite for add-on 2. Add-on 1 must therefore be installed as a condition for installing add-on 2. In this case, no conflict check is performed for the prerequisite component. To ensure that add-on 1 can still run, however, you need to make sure that add-on 2 contains the modification from add-on 1. This means that add-on 1 also needs to be installed in your system landscape, so that you can test the functionality of both add-ons.

You specify the prerequisite relationship using the import conditions in Software Delivery Assembler. When registering the package, select the Import Conditions tab page under Add-On Component and enter the release for add-on 1, and then select the option T (= True).

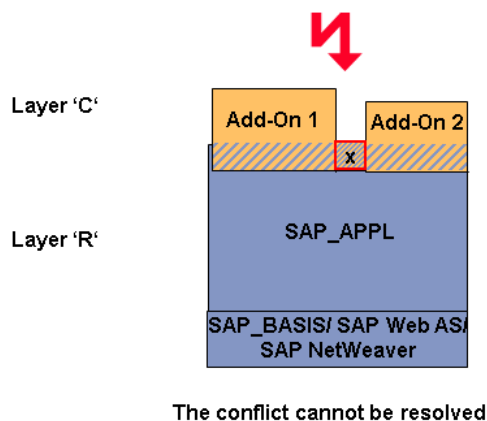


Conflicts with Add-On Components from the Same Layer

Add-On Installation Tool checks for conflicts with add-on components in the same layer.

Example: Add-on 1 and add-on 2 in layer 'C' both modify the same object in component SAP_APPL (in layer 'R'). Add-on 1 has already been installed. During installation of add-on 2, Add-On Installation Tool detects a conflict and aborts the installation process.

In this case, there is no way of resolving the conflict. It is not possible for both add-ons to be installed in one system.

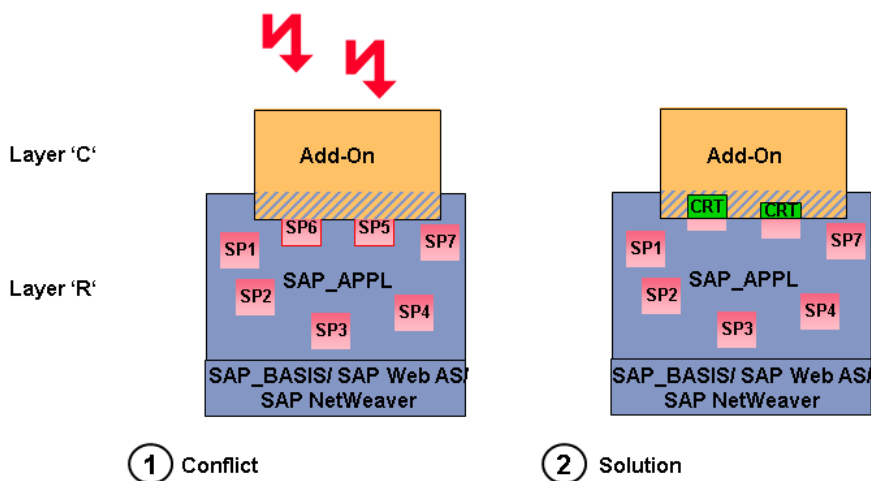


Conflicts with Support Packages in Lower Layers

Add-On Installation Tool checks for conflicts with Support Packages from software components in lower layers that exceed the import conditions, that is, with installed Support Packages whose level is higher than the prerequisite Support Package level in the import conditions. Add-On Installation Tool does not check for conflicts with Support Packages that are prerequisite in the import conditions.

Example: An add-on in layer 'C' modifies objects in component SAP_APPL (in layer 'R'). Four SAP_APPL Support Packages are prerequisites for the add-on. In one system, seven SAP_APPL Support Packages have already been imported. Support Packages 5 and 6 contain corrections to some of the objects modified by the add-on. When the add-on is installed in this system, Add-On Installation Tool detects conflicts with Support Packages 5 and 6. It then prompts the user to add a Conflict Resolution Transport to the queue.

If you created the add-on, you are responsible for resolving the conflict. You do this by creating Conflict Resolution Transport for Support Packages 5 and 6, or by creating a collective CRT for both of them. The procedure is described under [Creating a Conflict Resolution Transport \[page 120\]](#).



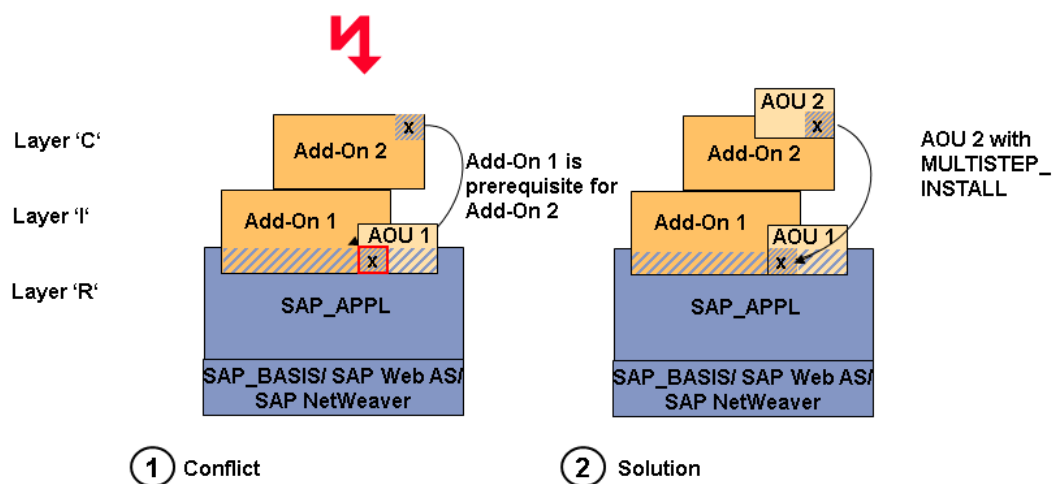
Conflicts with Add-On Components in Higher Layers

During add-on installation, Add-On Installation Tool checks for conflicts with add-on components in higher layers.

Example: Add-on 1 in layer 'I' and add-on 2 in layer 'C' modify the same object in component SAP_APPL (in layer 'R'). Both add-ons are installed in one system. SAP now creates an upgrade package (AOU) for add-on 1 in layer 'I'. This contains the modified object again. When add-on 1 is upgraded, Add-On Installation Tool detects conflicts with add-on 2 and aborts the upgrade package installation process, in order to prevent the modifications from add-on 2 from being overwritten.

If you created the add-on, you are responsible for resolving the conflict. You also do this by creating an upgrade package (AOU).

To do this, you need to assign the extended attribute MULTISTEP_INSTALL to your upgrade package in Software Delivery Assembler. This allows Add-On Installation Tool to install two upgrade packages in one installation queue at the same time. In this case, these are: AOU 1 for add-on 1 and AOU 2 for add-on 2. Normally, only one installation or one upgrade package is allowed. While registering the package, select the Extended Attributes tab page and select MULTISTEP_INSTALL. Then assign the value T (=True) to this attribute.

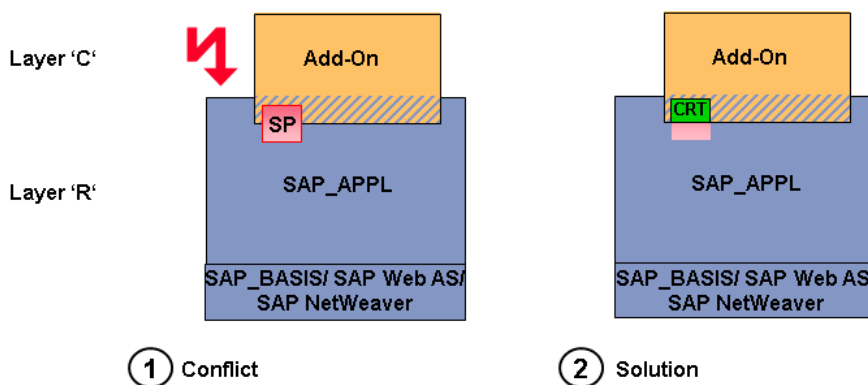


13.4.3 Conflicts When Importing Support Packages

Whenever Support Packages are imported, Support Package Manager checks for conflicts with add-on components from higher layers.

Example: An add-on from layer 'C' modifies an object from component SAP_APPL (from layer 'R'). If a SAP_APPL Support Package contains corrections to the modified object, Support Package Manager identifies a conflict and prompts the user to add a Conflict Resolution Transport to the queue.

If you created the add-on, you are responsible for resolving the conflict. You do this by creating a Conflict Resolution Transport for the Support Package. The procedure is described under [Creating a Conflict Resolution Transport \[page 120\]](#).



13.5 Examples: Attributes in Software Delivery Assembler

In addition to the attributes already generated automatically by the system it is possible to enhance additional attributes depending on the use case. In the following sections you will find examples for specific use cases:

- [Examples: Attributes for Add-On Uninstallations \[page 128\]](#)
- [Examples: Attributes for Enhancement Packages \[page 129\]](#)
- [Examples: Attributes for SAP HANA \[page 130\]](#)

13.5.1 Examples: Attributes for Add-On Uninstallations

The following attributes are relevant when determining whether add-ons can be uninstalled: `DEINSTALL_ALLOWED` and `DEINSTALL_PLUGIN`.

For more information about these attributes, see [Attributes for Uninstallations \[page 87\]](#).

Table 24: Examples of Attributes Set for Uninstallable Add-Ons

Attribute	Value	Comment
DEINSTALL_ALLOWED	1883223	This is the number of the global SAP Note. Add any add-on-specific information to your add-on documentation.
DEINSTALL_PLUGIN	/FINA/CL_FINA_DEINST	<p>The ABAP class must comply with the implementation instructions (see also Plug-In Interface for Add-Ons) [page 88].</p> <p>This attribute is optional. You can use it to implement a check class for your add-on.</p>
NEED_SPAM_LEVEL	55	<p>The lowest version of the SPAM/SAINT update required for uninstalling add-ons is 55. Always advise your customers, however, to use the highest available SPAM/SAINT version available in SAP Support Portal (a link to the download is in SAP Support Portal under https://help.sap.com/spmanager).</p> <p>We recommend using this attribute to ensure that your customers always have the correct version of the SPAM/SAINT update.</p>

You can also deliver the attribute at a later time in an attribute change package. You do this by setting the registration option *Post-Delivery with ACP*.

13.5.2 Examples: Attributes for Enhancement Packages

If the prerequisite software component versions of an add-on are violated when an enhancement package is installed, you can remove this violation by providing updated attributes for the add-on. You can deliver the new prerequisites using an attribute change package (ACP) for the add-on software component. You no longer need to request a vendor key.

Caution

The new import prerequisites cannot be delivered until they have been checked. The functions of the add-on must be capable of running on the enhancement package level. Any unchecked import prerequisites can produce unexpected errors.

Example

Component	Release	Value
EA-FINSERV	600	T
SAP_BASIS	700	T

Additional alternative import conditions that respect the enhancement package.

Component	Release	Value
EA-FINSERV	603	T
SAP_BASIS	701	T

13.5.3 Examples: Attributes for SAP HANA

The extended attribute `NEEDED_DBSYS` enables you to define database dependencies for an add-on. If this attribute is not set, the add-on can be used on any database. More specifically, this extended attribute makes it possible to develop SAP HANA-specific add-on and deliver them using *SAP HANA Transport for ABAP (HTA)* or *SAP HANA Transport Container (HTC)* (depending on the underlying SAP release).

For more information, see the documentation [Transport Scenarios for SAP HANA Content](#) in SAP Help Portal.

Attribute	Value	Comment
NEEDED_DBSYS	HDB	The package can be imported on SAP HANA. It can also be imported on any other database.
NEEDED_DBSYS	HDB:R	The package must be imported on SAP HANA. It cannot be imported on any other database.

13.6 CDs for Add-On Deliveries

This document contains information about the following topics:

- Structure of an installation CD/add-on upgrade CD
 - Structure of a CD with one add-on for different SAP releases
 - Structure of a CD with multiple add-ons for one SAP release
- How to create packed CD delivery data
- How to unpack SAR archives

Structure of an Installation CD/Add-On Upgrade CD

This structure applies to CDs containing one add-on for one SAP release.

Define the structure of the CD as follows:

- **DATA** (mandatory)
Save the SAR archive for the add-on package in question in this directory.
- **LANGUAGE** (optional)
You can save language-dependent data in this directory, if required.
- **PATCHES** (optional)
You can save the SAR archives for the add-on support packages in question in this directory, if required
- **DOCU** (optional)
You can save technical documentation about the import in this directory. (See also: [Template: Installation Guide \[page 69\]](#))
You can save the add-on-specific documentation in this directory.
- **README.ASC** (mandatory)
This file is a list of all files contains on the CD, with a short description.
Create the file **README.ASC**. Once you have completed this file, create the associated EBCDIC file on UNIX with the following command:

```
dd if=README.ASC of=README.EBC conv=ebcdic
```


You require the **dd** command to convert the file **README.ASC** to **README.EBC** for the platforms IBM eServer iSeries and IBM z/OS .

Note

The **dd** command is recognized on UNIX. On Windows, you must install the package *Windows Services for UNIX*.

Structure of a CD with One Add-On for Different SAP Releases

- Directory for SAP release 1
 - **DATA** (see above)
 - **(LANGUAGE)** (see above)
 - **(PATCHES)** (see above)
 - **(DOCU)** (see above)
- Directory for SAP release n
 - **DATA** (see above)
 - **(LANGUAGE)** (see above)
 - **(PATCHES)** (see above)
 - **(DOCU)** (see above)
- **README.ASC** (see above)
- **README.EBC** (see above)

Structure of a CD with Multiple Add-Ons for One SAP Release

- Directory for add-on 1
 - **DATA** (see above)
 - **(LANGUAGE)** (see above)

- (PATCHES) (see above)
- (DOCU) (see above)
- Directory for add-on n
 - DATA (see above)
 - (LANGUAGE) (see above)
 - (PATCHES) (see above)
 - (DOCU) (see above)
- README.ASC (see above)
- README.EBC (see above)

How to Create Packed CD Delivery Data

When you deliver an add-on on CD, you must place the delivery data itself in the CD directory DATA as a SAR archive.

Note

*.SAR archives can be created using the SAPCAR tool, which is located in every hardware directory (HPUX, AIX, and so on) on the kernel CD.

For more information about the SAPCAR tool, see SAP Note [212876](#) .

The SAR archive must meet the ISO standard 9660, level 2 for file and directory names:

- File and directory names can have no more than 31 characters.
- Use only uppercase letters, numerals, and underscores.

To create SAR archives for delivery packages, proceed as follows (this example is for UNIX):

1. Go to the transport directory:
cd /usr/sap/trans
2. Copy the *.PAT file from the directory EPS/out to the directory EPS/in:
cp EPS/out/<PAT file> EPS/in/<PAT file>
Here, the *.PAT file is the delivery package created using Software Delivery Assembler.
3. Use the following command to pack the data in the transport directory:
SAPCAR -cvf <name>.SAR EPS/in/<Pat file>
Replace <name> with a name that matches your add-on package and meets the ISO standard mentioned above.

How to Unpack SAR Archives

On Unix, the command for unpacking SAP archives manually is `SAPCAR -xvf /usr/sap/trans/<name>.SAR`.

To unpack SAR archives, you can also use the function ► [Load Packages](#) ► [From Front End](#) ► in Support Package Manager or Add-On Installation Tool . This also uploads the SAR archive and unpacks it at the same time. Choose one of the following menu options:

Table 25:

Import Tool	Menu Option
Support Package Manager	► Support Package ► Load Packages ► From Front End ►.
Add-On Installation Tool	► Installation Package ► Load Packages ► From Front End ►.

13.7 Troubleshooting

You can create messages for SAP Add-On Assembly Kit in SAP Support Portal under <https://support.sap.com/message> ►. Use the component XX-PROJ-AAK.


13.8 Links in SAP Help Portal

The following guides you to more detailed information in the different releases in SAP Help Portal under <https://help.sap.com>






Table 26:

Topic Area	Release	Further Information in SAP Help Portal
Change and Transport System (Transport Management System/ language transports)	SAP NW 7.4.	Change and Transport System ► Transport Management System (BC-CTS-TMS)/Language Transport (BC-CTS-LAN) ►
	SAP NW 7.3 EHP1	Change and Transport System ► Transport Management System (BC-CTS-TMS)/Language Transport (BC-CTS-LAN) ►
	SAP NW 7.3.	Change and Transport System ► Transport Management System (BC-CTS-TMS)/Language Transport (BC-CTS-LAN) ►

Topic Area	Release	Further Information in SAP Help Portal
	SAP NW 7.0 EHP 2	Change and Transport System ► Transport Management System (BC-CTS-TMS)/Language Transport (BC-CTS-LAN) ►
	SAP NW 7.0 EHP1	Change and Transport System ► Transport Management System (BC-CTS-TMS)/Language Transport (BC-CTS-LAN) ►
	SAP NW 7.0.	Change and Transport System ► Transport Management System (BC-CTS-TMS)/Language Transport (BC-CTS-LAN) ►
Software logistics/software maintenance (Note that you can display the current version of the documentation about software maintenance tools using the help function in the pushbutton bar on the initial screen of the tool in question.)	SAP NW 7.4.	Software Logistics
	SAP NW 7.3 EHP 1	Software Logistics
	SAP NW 7.3.	Software Logistics
	SAP NW 7.0 EHP 2	Software Logistics
	SAP NW 7.0 EHP 1	Software Logistics
	SAP NW 7.0.	Software Logistics
Documentation and translation tools	SAP NW 7.4.	Services for Information Developers and Translators
	SAP NW 7.3 EHP 1	Services for Information Developers and Translators
	SAP NW 7.3.	Services for Information Developers and Translators
	SAP NW 7.0 EHP 2	Documentation and Translation Tools (BC-DOC)
	SAP NW 7.0 EHP 1	Documentation and Translation Tools (BC-DOC)
	SAP NW 7.0.	Documentation and Translation Tools (BC-DOC)
ABAP programming basics	SAP NW 7.4.	Application Development on AS ABAP
	SAP NW 7.3 EHP 1	Application Development on AS ABAP
	SAP NW 7.3.	Application Development on AS ABAP
	SAP NW 7.0 EHP 2	ABAP Technology

Topic Area	Release	Further Information in SAP Help Portal
	SAP NW 7.0 EHP 1	ABAP Technology
	SAP NW 7.0.	ABAP Technology
Package concept	SAP NW 7.4.	Package Builder and ABAP Package Concept
	SAP NW 7.3 EHP 1	Package Builder and ABAP Package Concept
	SAP NW 7.3.	Package Builder
	SAP NW 7.0 EHP 2	Package Builder and ABAP Package Concept
	SAP NW 7.0 EHP 1	Package Builder
	SAP NW 7.0.	Package Builder
BC Sets and IMG enhancements	SAP NW 7.4.	<ul style="list-style-type: none"> • Business Configuration Sets (BC-CUS)  • IMG Enhancement 
	SAP NW 7.3 EHP 1	<ul style="list-style-type: none"> • Business Configuration Sets (BC-CUS)  • IMG Enhancement 
	SAP NW 7.3.	<ul style="list-style-type: none"> • Business Configuration Sets (BC-CUS)  • IMG Enhancement 
	SAP NW 7.0 EHP 2	<ul style="list-style-type: none"> • Business Configuration Sets (BC-CUS) • IMG Enhancement
	SAP NW 7.0 EHP 1	<ul style="list-style-type: none"> • Business Configuration Sets (BC-CUS) • IMG Enhancement
	SAP NW 7.0.	<ul style="list-style-type: none"> • Business Configuration Sets (BC-CUS) • IMG Enhancement
Roles and authorizations	SAP NW 7.4.	User and Role Administration of Application Server ABAP
	SAP NW 7.3 EHP 1	User and Role Administration of Application Server ABAP
	SAP NW 7.3.	User and Role Administration of Application Server ABAP
	SAP NW 7.0 EHP 2	User and Role Administration of AS ABAP

Topic Area	Release	Further Information in SAP Help Portal
	SAP NW 7.0 EHP 1	User and Role Administration of AS ABAP
	SAP NW 7.0.	User and Role Administration of AS ABAP
Delivery class of tables	SAP NW 7.4.	Delivery Class of Database Tables
	SAP NW 7.3 EHP 1	Delivery Class
	SAP NW 7.3.	Delivery Class
	SAP NW 7.0 EHP 2	Delivery Class
	SAP NW 7.0 EHP 1	Delivery Class
	SAP NW 7.0.	Delivery Class
The Modification Assistant	SAP NW 7.4.	The Modification Assistant
	SAP NW 7.3 EHP 1	The Modification Assistant
	SAP NW 7.3.	The Modification Assistant
	SAP NW 7.0 EHP 2	The Modification Assistant
	SAP NW 7.0 EHP 1	The Modification Assistant
	SAP NW 7.0.	The Modification Assistant
Changing the SAP standard (BC)	SAP NW 7.4.	Changing the SAP Standard (BC)
	SAP NW 7.3 EHP 1	Changing the SAP Standard (BC)
	SAP NW 7.3.	Changing the SAP Standard (BC)
	SAP NW 7.0 EHP 2	Changing the SAP Standard (BC)
	SAP NW 7.0 EHP 1	Changing the SAP Standard (BC)
	SAP NW 7.0.	Changing the SAP Standard (BC)
Enhancement Framework	SAP NW 7.4.	Enhancement Framework
	SAP NW 7.3 EHP 1	Enhancement Framework
	SAP NW 7.3.	Enhancement Framework
	SAP NW 7.0 EHP 2	Enhancement Framework
	SAP NW 7.0 EHP 1	Enhancement Framework

Topic Area	Release	Further Information in SAP Help Portal
	SAP NW 7.0.	Enhancement Framework
Enhancement spots	SAP NW 7.4.	Enhancement Spots
	SAP NW 7.3 EHP 1	Enhancement Spots
	SAP NW 7.3.	Enhancement Spots
	SAP NW 7.0 EHP 2	Enhancement Spots
	SAP NW 7.0 EHP 1	Enhancement Spots
	SAP NW 7.0.	Enhancement Spots
Append structures	SAP NW 7.4.	Append Structures
	SAP NW 7.3 EHP 1	Append Structures 
	SAP NW 7.3.	Append Structures 
	SAP NW 7.0 EHP 2	Append Structures 
	SAP NW 7.0 EHP 1	Append Structures 
	SAP NW 7.0.	Append Structures 

13.9 Terminology

The following contains some of the most important terms used in this documentation:

- **add-on**
An additional software component that can be installed in the system Add-ons can be specific to an industry, country, or enterprise, or can be customer or partner projects.
They are installed using a delivery package imported using Add-On Installation Tool. The delivery package combines multiple transport request and contains the development objects of the add-on.
- **add-on delta upgrade**
Upgrade of the add-on release without upgrading the SAP system
- **add-on exchange upgrade**
Upgrade of the add-on release while upgrading the SAP system
- **add-on exchange package**
Used in add-on exchange upgrades. Add-on exchange packages are available from SAP NetWeaver 7.0.
- **add-on release**
Release (version) of the add-on
- **add-on support package**

Support package for an add-on- Depending on the underlying SAP release, the associated SAP release is either **CSP** (for component support packages from SAP Web AS 6.40) or **AOP** (for add-on support packages up to SAP Web AS 6.20).

- **change request**
Information container used in Transport Organizer to create and manage all changes made to repository objects and customizing settings as part of a development project.
- **change list**
List of all new and modified objects in Software Delivery Composer relevant for a specific delivery.
This includes:
 - Change piece list
 - Supplement change piece list
 - Support package
 - CRT
- **Change piece list**
In Software Delivery Composer, list of new, modified, and deleted objects in the delivery component for an add-on upgrade
- **application component**
Software component that is not part of the SAP NetWeaver platform and that cannot be installed retroactively in an SAP system (such as SAP_APPL or SAP_HR)
- **delivery**
Set of software objects delivered together
A delivery consists of a single delivery component. Deliveries are used to ship installations, upgrades, support packages, and conflict resolution transports.
The dependencies that can exist between different deliveries mean that only certain combinations of deliveries produce a working system state.
- **delivery request**
In Software Delivery Composer, an information container for creating all transport objects and software objects relevant for a specific delivery.
This includes:
 - Change list
 - Component piece list
 - Supplement component piece list
 - Exchange component piece list
 - Exclusive lists
- **delivery component**
Set of software objects. A delivery component comprises the following objects:
 - Software object of the modifying software component
 - Modified objects from a different software component
- **delivery type**
The delivery type determines how the delivery is created and the delivery requests required. Software Delivery Composer distinguishes the following delivery types when creating the delivery:
 - Installation/upgrade
 - Conflict resolution transport
 - Support package
- **exchange component piece list**
List of all objects in the current delivery component for an add-on exchange upgrade.

The exchange component piece list covers the current component piece list and the component piece lists of the predecessor versions supported as source releases.

The package type used to register the exchange component piece list in Software Delivery Assembler is distinguished as follows in accordance with the underlying SAP release:

- From SAP NetWeaver 7.0:
Add-on exchange package (AOX)
- Up to and including SAP Web AS 6.40:
Add-on upgrade package (AOU) with import attribute 'CREATE_FULLTASK=T'
- Conflict resolution transport (CRT)
 1. In Software Delivery Composer, list of adjusted software objects for a modified software component for a conflict resolution transport. A conflict resolution transport can also contain corrected software objects for the delivery component.
 2. Package type in Software Delivery Assembler. Restores add-on-specific modifications that were overwritten by SAP support packages.
- exclusive list
List of objects in a software component that are not shipped in the current delivery.
- Main component (up to and including SAP Web AS 6.10):
Software component in the SAP system that is not flagged as an add-on, for example SAP_BASIS or SAP_HR. In later releases, the main components are divided into application components and Web AS/NetWeaver components.
- component piece list
In Software Delivery Composer, a list of all objects in the current delivery component for an add-on installation.
The component piece list covers the current change piece list and the component piece list of the predecessor version.
- modification
Customer-specific change to SAP repository objects.
In the case of changes made by SAP, modified SAP repository objects are identified in the import and proposed for modification adjustment.
In the case of SAP Add-On Assembly Kit, modification is defined as any changes made by add-on developers to objects in the standard SAP system.
- namespace
A name granted exclusively by SAP to SAP customers and SAP partners that enables them to develop their own components and products based on SAP applications without the risk of naming conflicts (which can occur, for example, when making deliveries to third-party SAP systems or when importing third-party products). Namespaces can be requested in SAP Support Portal.
Objects in ABAP Workbench are assigned to a namespace by prefixing the object name with the reserved namespace prefix. The namespace name starts and ends with the delimiter character "/" and can have a maximum of 10 characters.
A namespace is also required for the add-on software component and when creating deliveries using SAP Add-On Assembly Kit.
- namespace role
Defines the origin of the development objects and the type of changes that can be made to objects in this namespace.
The role **P**(for producer) can be chosen only if the development license is known. New objects can be created. The role **C** (for consumer) can be used to make repairs if a valid repair license is entered. The role is always **C** in systems that are given the namespace. The functions developed in this namespace can be used. Development work, however, is not possible.

- package type
Type of an importable package.
Software Delivery Assembler converts delivery requests to importable packages. You can use SAP Add-On Assembly Kit to create the following package types:
 - Add-on installation package (AOI)
 - Add-on upgrade package (AOU)
 - Add-on support package (CSP (or AOP))
 - Conflict resolution transport (CRT)
 - Add-on exchange package (AOX)
- SAP standard
Standard software delivered by SAP. This includes all components in an SAP delivery (for example SAP ERP) with all their components.
- SAP system upgrade
Release upgrade of SAP software performed using the SAP upgrade tool Software Update Manager (SUM). When performed, an SAP system upgrade usually involves a switch to a new SAP NetWeaver release.
- software component
Set of software objects bundled in packages and which can only be delivered together. A software component usually exists in several different releases. Upgrades make it possible to switch from one release to another. Support packages can be create for each software component.
- software component version
Version of a software component identified by an add-on ID and add-on release.
- support package
 1. In Software Delivery Composer, a list of corrected software objects in the delivery component for an add-on support package.
 2. Package type in Software Delivery Assembler. Contains quality enhancements for a specific software component.
- supplement change piece list
In Software Delivery Composer, a list of new and modified software objects in the delivery component for an upgrade supplement (an appendix to the SAP system upgrade). Supplement change piece lists are available up to SAP Web AS 6.10.

Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

Accessibility

The information contained in the SAP documentation represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of wilful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

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.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP's gross negligence or willful misconduct. All links are categorized for transparency (see: <http://help.sap.com/disclaimer>).



**go.sap.com/registration/
contact.html**

© 2016 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.