



User Guide | PUBLIC

SAP Add-On Assembly Kit 5.0 Support Package 4
2021-06-21

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	New Features in Version 5.0.	9
3	Process Flow: SAP Add-On Assembly Kit.	20
4	Background Information: Add-Ons	23
5	Software Component Layers.	25
5.1	Software Component Hierarchy from SAP Web AS 6.40.	27
6	Defining the Delivery Strategy	29
7	Defining the Namespaces	31
7.1	Rules for Namespaces.	32
7.2	Reserving a namespace.	33
7.3	Add-On Software Component	33
7.4	Add-On Release Name	34
7.5	Naming Conventions for Delivery Requests	36
8	Setup of the System Landscape and Systems	38
8.1	System Landscape for Add-On Development	38
	Setting Up a Development Landscape for the First Add-On Release.	39
	Setting Up a Development Landscape for Further SAP Releases	40
8.2	System Landscape for Add-On Maintenance	41
	Setting Up A Maintenance System Using a System Copy	42
	Setting Up a Maintenance System Using a Transport of Copies.	43
8.3	Developing Multiple Add-Ons in a System Landscape	44
8.4	Client Layout and Transport Paths.	46
8.5	Installing SAP Add-On Assembly Kit	47
8.6	Configuring the Development System and Consolidation System	48
	Entering a Namespace and Defining a Namespace Role.	48
	Creating or Updating an Add-On Software Component and Add-On Release Name	49
	Setting the System Change Option	51
	Configuring Parameters for the Transport Control Program tp.	52
8.7	Configuring the Maintenance Systems	53
8.8	Upgrading the Development/Maintenance Landscape	54

8.9	Test Systems.	54
9	Add-On Development	56
9.1	Creating and Assigning Packages.	56
9.2	Rules for Add-On Development	57
	Rules for Add-On Development on Multiple SAP Releases.	58
9.3	Additional Information About Enhancements and Modifications.	59
9.4	Documentation and Translation	60
10	Delivering the Add-On Software.	62
10.1	Package Types.	63
10.2	Final Assembly System.	67
10.3	Delivery Creation.	67
	Assembling a Delivery with SDC.	68
	Creating a Delivery Package with SDA.	69
	Testing the Delivery.	71
	Removing Errors in the Delivery.	72
	Post-Delivery of Attributes.	74
	Finishing the Creation of the Delivery.	75
	Object List Checks.	76
	Migrating Packages.	86
10.4	Creating Add-On Installation Packages	88
10.5	Providing a Delivery	88
	Template: Installation Guide	89
10.6	Delivering Languages Translated Retroactively.	91
	Delivering Languages Using a Add-On Language Package.	92
	Delivering Languages Using a Support Package.	94
11	Maintenance and Upgrade.	96
11.1	Rules for Add-On Maintenance	97
11.2	Creating Maintenance Packages	97
	Creating Add-On Support Packages	97
11.3	Creating Add-On Upgrade Packages (Upgrading the Add-On Software)	98
11.4	Add-On Behavior in SAP System Upgrades	99
	Updating Add-Ons in SAP System Upgrades.	100
11.5	End of Maintenance	102
12	Add-On Uninstallation.	103
12.1	Checklist: Development Aspects.	104
12.2	Checklist: Landscape Aspects.	105
12.3	Checklist: System Aspects.	106
12.4	Checklist: Test Aspects.	107
12.5	Checking Add-Ons for Uninstallability.	108

12.6	Simulating an Add-On Uninstallation.	110
	Critical Objects.	112
12.7	Attributes for Uninstallations.	112
12.8	Plug-In Interface for Add-Ons.	113
12.9	Handling Object Types.	118
13	Additional Information	119
13.1	Compatibility of SAP_BASIS Releases.	119
13.2	Overview: Import Tools.	121
13.3	Modifications and Their Consequences	121
	Setting Up a Development Landscape for Modifying Add-On.	122
	Maintenance for a Modifying Add-On.	124
	Rules for Developing a Modifying Add-On	125
	Creating Conflict Resolution Transports	126
13.4	Conflicts	128
	Background Information: Conflict Check	128
	Conflicts When Installing/Upgrading an Add-On.	129
	Conflicts when Importing Support Packages	133
13.5	Examples: Attributes in Software Delivery Assembler.	134
	Examples: Attributes for Add-On Uninstallations.	134
	Examples: Attributes for Enhancement Packages.	135
	Examples: Attributes for SAP HANA.	135
13.6	CDs for Add-On Deliveries.	136
13.7	Troubleshooting	138
13.8	Further Documentation.	139
13.9	Terminology.	148

1 SAP Add-On Assembly Kit

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.

i Note

SAP Add-On Assembly Kit 5.0 is available for SAP components - based on SAP NetWeaver 7.0 and higher SAP NetWeaver releases - and can only be used for ABAP developments.

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 \[page 121\]](#).

Prerequisites

- To be able to work with SAP Add-On Assembly Kit, you need certain [authorizations \[page 6\]](#).
- You should have knowledge of the following topics:
 - Installation of SAP systems
 - System copy
 - Setting up the system landscape (including Change and Transport System)
 - ABAP programming basics
 - ABAP software maintenance
 - Documentation and translation tools

Links and paths to further information about these topics can be found under [Further Documentation \[page 139\]](#).

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:

Overview of Roles

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

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

i Note

Before you start working with SAP Add-On Assembly Kit, make sure you have the latest version of this document. You can find it in SAP Help Portal under <https://help.sap.com/aak>. For more information, also see SAP Note [3066216](#). It contains documentation changes that were required after the last publication.

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

Structure

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

1. [Defining the Delivery Strategy \[page 29\]](#)

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 uninstallable?

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

2. [Defining the Namespaces \[page 31\]](#)

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 and Systems \[page 38\]](#)

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 \[page 56\]](#)


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 Software \[page 62\]](#)
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 \[page 96\]](#)
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 \[page 103\]](#)
SAP Add-On Assembly Kit contains instructions and checks that verify that your add-on can be uninstalled.

Overview of SAP Notes

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

Note Number	Short Description
16466 	Customer namespace for SAP objects
33040 	Options in Exchange Upgrade with Add-On
70228 	Add-Ons: Prerequisites and Upgrade Planning
104010 	Restrictions on development in namespaces
105132 	Reserving namespaces from R/3 Release 4.0
195442 	Language imports and support packages
212876 	New archiving tool SAPCAR
215178 	Software Delivery Composer: Object list checks
395083 	FAQ Namespace for authorizations, authorization objects and fields, and profiles
567695 	Password required to import Add-On Package
870407 	SDC: Object types that must not be delivered
1883223 	Note on general add-on uninstallations
2011192 	Uninstalling ABAP add-ons
2179441 	Installation/upgrade of SAP Add-On Assembly Kit 5.00
2308014 	Conversions to SAP S/4HANA On-Premise with 3rd Party / non-SAP Add-Ons

Note Number	Short Description
3066216 	Collective SAP Note for documentation changes: SAP Add-On Assembly Kit

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 New Features in Version 5.0

This section provides an overview of the functional changes and new features in SAP Add-On Assembly Kit from version 5.0.

General Information

- Support Package 4 of SAP Add-On Assembly Kit 5.0 requires SPAM/SAINT update 68 or higher.
- 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.



What's New

Title	Description	Type	Available from Version	Affected Tool	SPAM Update (SL Toolset)
Documentation: Introduction of a composite SAP Note for documentation changes	SAP Note 3066216  contains all changes and corrections to the documentation made necessary after publication of the last version of SAP Add-On Assembly Kit.	New		Documentation	SPAM update 0078
New delivery type: <i>Pilot</i>	Pilot deliveries are used to test functional changes in a test system before they are made available in production systems.	New	AAK 5.0 SP4	Software Delivery Composer	

Title	Description	Type	Available from Version	Affected Tool	SPAM Update (SL Toolset)
New expanded attribute: PREPACKAGE	<p>This enables you to flag delivery packages as pre-deliveries.</p> <div> <p>i Note</p> <p>Use this attribute only for delivery packages of type CRT or CSP.</p> </div> <p>SDA sets the time stamp of the registration time automatically as the attribute value.</p> <p>Pre-deliveries cannot be released for customers. They can be imported into partner systems if the import of pre-deliveries is released in the relevant system. To do this, run the report RSSET_PREPACK_4_CUV. Note that you should run this report in test systems only. If the import of pre-deliveries has not been released, the SL tools (SPAM, SAINT, SUM) reject the import.</p>	New	AAK 5.0 SP4	Software Delivery Assembler	
Expanded preconfigured selection conditions for selecting change requests	The time stamp now contains the date as well as an exact time.	Changed	AAK 5.0 SP3	Software Delivery Composer	
Flagged change requests can now be excluded from the change list using a mass function.		Changed	AAK 5.0 SP3	Software Delivery Composer	
The maximum number of alternative import conditions has been increased from three to ten.		Changed	AAK 5.0 SP3	Software Delivery Composer	
The maximum size of two GB for a package has been removed.		Deleted	AAK 5.0 SP3	Software Delivery Composer	

Title	Description	Type	Available from Version	Affected Tool	SPAM Update (SL Toolset)
The package type Add-on Exchange Package is now only partially supported.	From source release SAP_BASIS 751, it is no longer mandatory to create an exchange component piece list. More information: Package Types [page 63]	Changed	AAK 5.0 SP3	Software Delivery Assembler	
Revision of results display for object list checks.	<ul style="list-style-type: none"> Revision of the sorting (sequence and stability) Improved tree structure stability (avoidance of unintentional scrolling after node expansion and actions on nodes) Simplified result node selection and implicit selection of all subnodes Simplified comment function: Now only mandatory when <i>setting entries to "Done"</i>. New text is added to existing text (always when executing the action <i>Set Entries to 'Done'</i> and when using the comments function on the higher node following a corresponding response on the selection screen). 	Changed	AAK 5.0 SP2	Software Delivery Composer	
Delivery-specific option for expanding the object list check <i>Add-On Uninstallation: Non-Deletable Objects</i>		Changed	AAK 5.0 SP2	Software Delivery Composer	
Optimization of check <i>Generic Checks and Automatic Corrections</i>	This reduces the number of relevant messages.	Changed	AAK 5.0 SP2	Software Delivery Composer	
New expanded attribute: <code>NEEDED_UPGRADE</code> is available.	This expanded attribute always has to be set when the package import requires a (simultaneous) installation or upgrade of another software component version.	New	AAK 5.0 SP2	Software Delivery Assembler	
Confirmed deliveries can be reopened		New	AAK 5.0 SP1	Software Delivery Composer	

Title	Description	Type	Available from Version	Affected Tool	SPAM Update (SL Toolset)
The PAT file of a package can now be downloaded as an SAR file.	The downloaded SAR file can be uploaded directly using transactions <i>SPAM</i> and <i>SAINT</i> .	New	AAK 5.0 SP1	Software Delivery Assembler	
New object list checks	<ul style="list-style-type: none"> <i>Add-On Uninstallation: Non-Deletable Objects</i> Add-on uninstallations 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 such objects have been delivered, the add-on can only be uninstalled if a plug-in class is defined for this add-on (see Plug-In Interface for Add-Ons [page 113]). <i>Generic Checks and Automatic Corrections</i> 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. 	New	AAK 5.0	Software Delivery Composer	
Comment function in results list of object list checks		New	AAK 5.0	Software Delivery Composer	
New roles introduced		New	AAK 5.0	Software Delivery Composer	
SDA online documentation update		Changed	AAK 5.0	Software Delivery Assembler	

Title	Description	Type	Available from Version	Affected Tool	SPAM Update (SL Toolset)
New expanded attribute: DEINSTALL_ALLOWED	Only for the package types A01, A0U, and A0X: This software component version can be uninstalled. Also see the information about Add-On Uninstallation [page 103] . 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.	New	AAK 5.0	Software Delivery Assembler	

Title	Description	Type	Available from Version	Affected Tool	SPAM Update (SL Toolset)
New expanded attribute: DEINSTALL_PLUGIN	<p>Only for the package types AOI, AOU, and AOX: 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:</p> <ul style="list-style-type: none"> • 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. <p>Also see the information under Attributes for Uninstallations [page 112]. 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.</p>	New	AAK 5.0	Software Delivery Assembler	
New expanded attribute: KERNEL_VERSION	From SAP Web AS 6.10: Specifies the kernel version required to import the package, for example: 620/00123.	New	AAK 5.0	Software Delivery Assembler	

Title	Description	Type	Available from Version	Affected Tool	SPAM Update (SL Toolset)
New expanded attribute: <code>LANGUAGE_BY_SP</code>	<p>This attribute can be used for packages of the type <code>AOU</code> and <code>CSP</code> as of SAP NetWeaver 7.0. If new languages are delivered with a package with one of these types, they can be registered as new installed languages in the import to a target system by setting the attribute <code>LANGUAGE_BY_SP</code>. The languages specified in the attribute must already exist in the attribute <code>LANGUAGE</code>. The values for this attribute must be specified using the two-character ISO language key.</p> <div> <p>❖ Example</p> <p><code>LANGUAGE_BY_SP ISO-FRJA</code></p> <p>This package delivers the languages FR (French) and JA (Japanese).</p> </div>	New	AAK 5.0	Software Delivery Assembler	

Title	Description	Type	Available from Version	Affected Tool	SPAM Update (SL Toolset)
New expanded attribute: NEEDED_DBSYS	<p>Applies to packages of the type AOI, AOU, and AOX as of SAP NetWeaver 7.0. Specifies a list of database systems where the add-on can be installed.</p> <p>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.</p> <p>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.</p> <p>The current possible values are:</p> <ul style="list-style-type: none"> • 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 <div> <p>❖ Example</p> <p>The package is to be imported on one of the following database systems: NEEDED_DBSYS HDB, INF</p> </div>	New	AAK 5.0	Software Delivery Assembler	

Title	Description	Type	Available from Version	Affected Tool	SPAM Update (SL Toolset)
	<p>The package must be imported on one of the following two database systems: NEEDED_DBSYS HDB, INF:R</p>				
New expanded attribute: R3TRANS_VERSION	<p>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 zeroes.</p> <p>❖ Example 20061025</p>	New	AAK 5.0	Software Delivery Assembler	
New expanded attribute: TP_VERSION	<p>Specifies the minimum version of the transport program tp required to import the package. Specify the version in the following format: <KERNEL_VERSION>/<VERSION>.</p>	New	AAK 5.0	Software Delivery Assembler	

Title	Description	Type	Available from Version	Affected Tool	SPAM Update (SL Toolset)
New expanded attribute: UPG_KEY_REQUEST	<p>For package types AOU through SAP Web AS 6.40, AOX as of 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.</p> <p>Possible value: T = True</p>	New	AAK 5.0	Software Delivery Assembler	
The new package type Attribute Change Packages (ACP) makes follow-up deliveries of change attributes possible.	<p>The package type Attribute Change Package 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 <i>Post-Delivery (with ACP)</i>, 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.</p>	New	AAK 5.0	Software Delivery Assembler	
Modified tab order for package registration		Changed	AAK 5.0	Software Delivery Assembler	

Title	Description	Type	Available from Version	Affected Tool	SPAM Update (SL Toolset)
	Modified display and editing of the attributes and import conditions	Changed	AAK 5.0	Software Delivery Assembler	
	New roles introduced	New	AAK 5.0	Software Delivery Assembler	

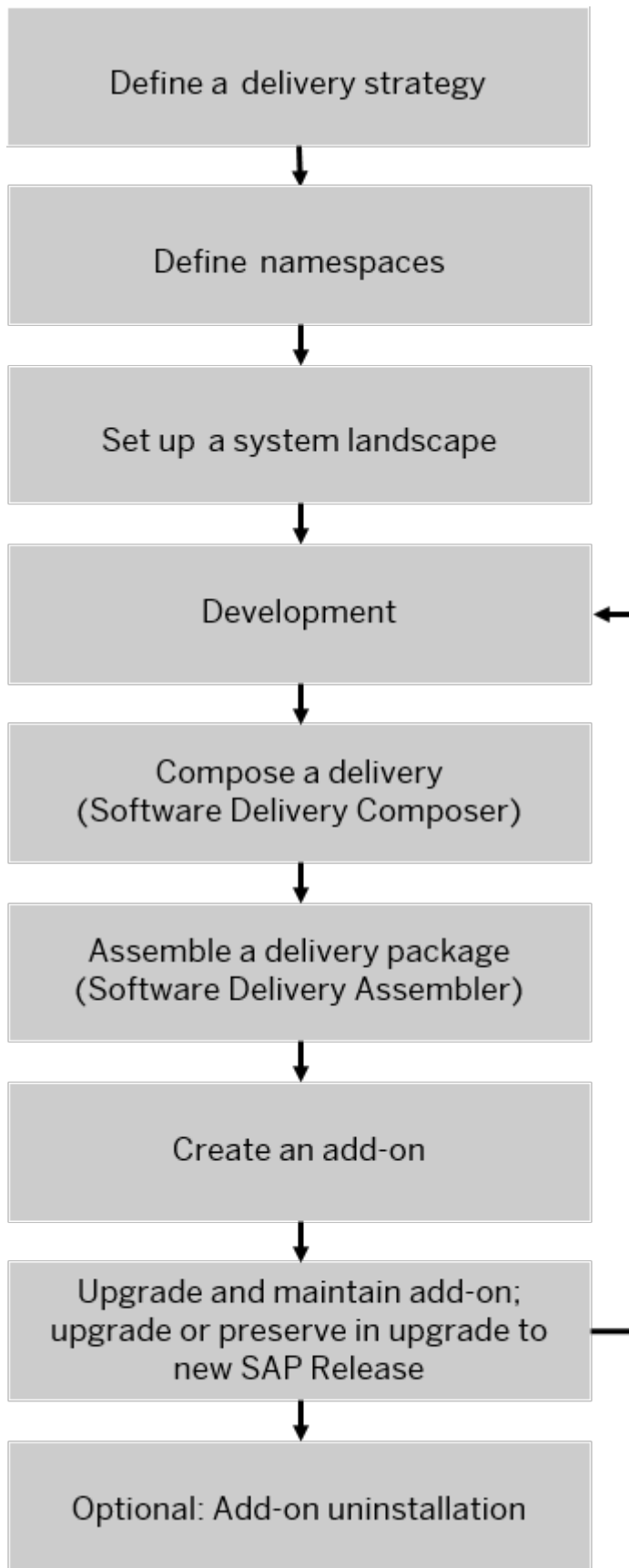
Documentation

The new and updated functions in SAP Add-On Assembly Kit 5.0 and the support packages have been included in the documentation.

3 Process Flow: SAP Add-On Assembly Kit

This is a fixed process for creating an add-on with the SAP Add-On Assembly Kit. Here you will find an overview of the individual steps.

The following figure illustrates the process flow for using the SAP Add-On Assembly Kit to create add-ons:



Creating Add-On with the SAP Add-On Assembly Kit

For more information about the individual steps, see the following links:

1. [Defining the Delivery Strategy \[page 29\]](#)
2. [Defining the Namespaces \[page 31\]](#) (for development objects and the add-on software component)
3. [Setup of the System Landscape and Systems \[page 38\]](#)
4. [Add-On Development \[page 56\]](#)
5. [Delivering the Add-On Software \[page 62\]](#)
6. [Maintenance and Upgrade \[page 96\]](#)
7. [Add-On Uninstallation \[page 103\]](#)

4 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.

Objects of an add-on

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 \[page 63\]](#):

- 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.

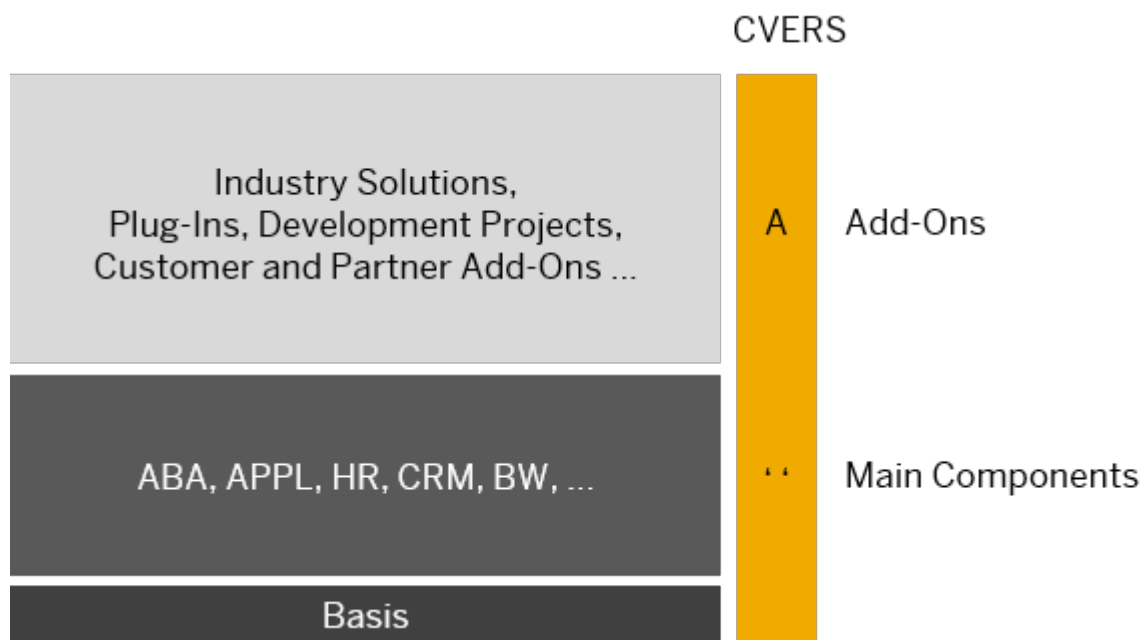
5 Software Component Layers

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.



Software Component Layers to SAP Web AS 6.10

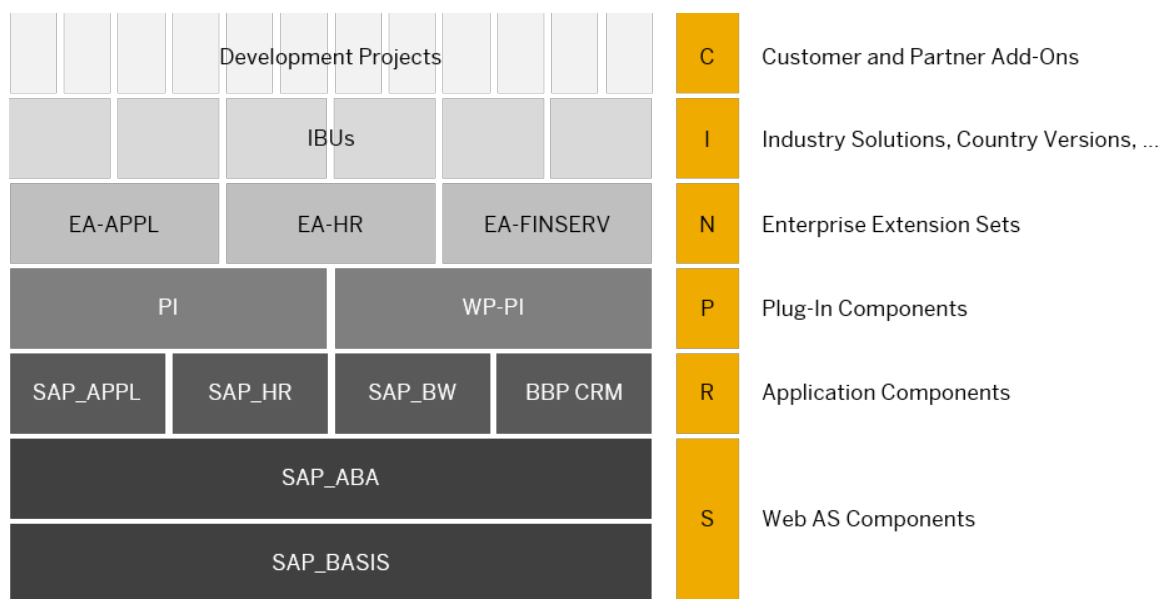
Software Component Hierarchy from SAP Web AS 6.20

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').

Note

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



Layers of the Software Component Hierarchy in SAP Web AS 6.20 (Example)

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.

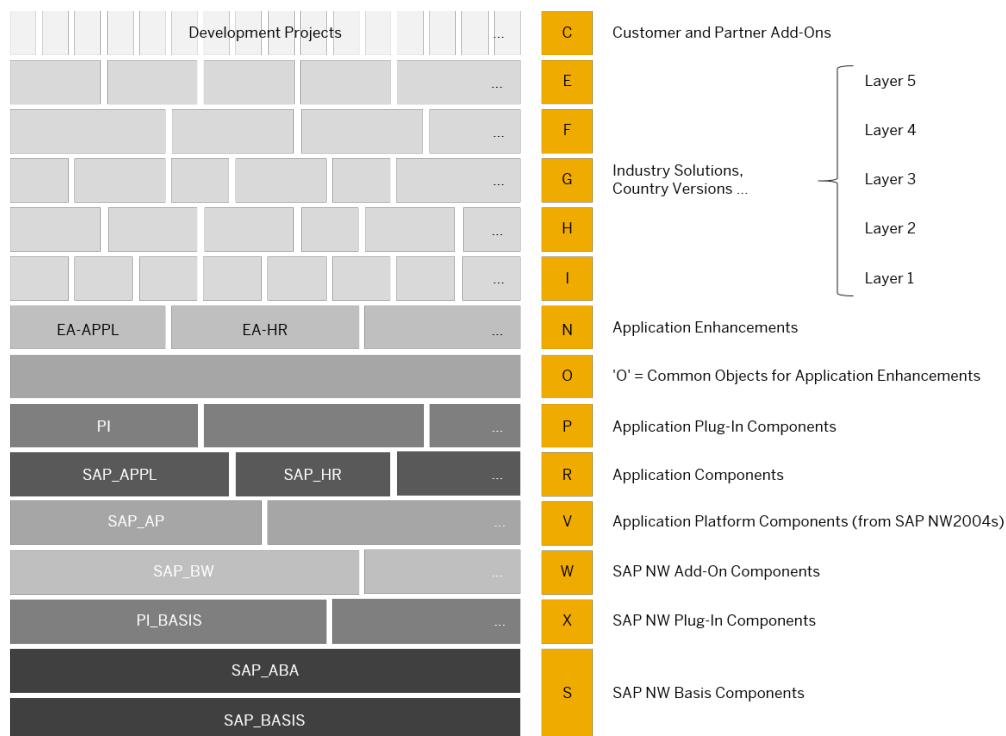
Optional Software Components (Add-On Components)

- 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.

5.1 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.



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)
These are the components of the NetWeaver application platform, such as SAP_AP. (From SAP NetWeaver 7.0)

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.

i Note

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

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 \[page 121\]](#) and [Conflicts \[page 128\]](#).

→ 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 does your upgrade strategy look like?

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 sections [Add-On Behavior in SAP System Upgrades \[page 99\]](#) and [End of Maintenance \[page 102\]](#).

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

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.

i 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.

Related Information

[Rules for Namespaces \[page 32\]](#)

[Reserving a namespace \[page 33\]](#)

[Add-On Software Component \[page 33\]](#)

[Add-On Release Name \[page 34\]](#)

[Naming Conventions for Delivery Requests \[page 36\]](#)

7.1 Rules for Namespaces

When defining namespaces, you need to keep several rules in mind.

General Rules

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

→ Recommendation

We recommend that you choose namespaces with at least five digits, since smaller namespaces are assigned internally at SAP.

❖ Example

`/ABCDEF /`

- Do not use a namespace that has already been reserved for SAP developments:
 - `/*SAP*/`
 - `/IS*/`
 - `/HR<two places>/`
 - `/P<two places>/`
- Uppercase letters and digits are allowed

Additional Information for Namespaces for the Add-On Software Component

To ensure that the add-on software [component \[page 33\]](#) is unique, the name of the add-on software component is automatically derived from the namespace reserved for the add-on software component. When you create an outbound delivery, the namespace prefix of the delivery name is used as the add-on software component.

❖ Example

If the namespace for the add-on software component is `/ABCDEF /`, then your add-on software component is `ABCDEF`.

Since the name of the add-on software component is visible and is also occurs in the name of the importable package, you should decide if you want to reserve a dedicated namespace for the add-on software component. Your add-on will then be known by this namespace.

You can also use the namespace for development objects for the add-on software component. If you have reserved multiple namespaces, you can select one namespaces to be used to derive the name of your add-on software component.

❖ Example

If you have reserved the namespaces `/ABCDEF1/`, `/ABCDEF2/` and `/ABCDEF3/` then your add-on software component can be named `ABCDEF2`.

7.2 Reserving a namespace

You need to create one namespace for the development objects and another namespace for the add-on software component (if this namespace is not identical to the namespace for the development objects).

You can reserve namespaces in SAP Support Portal under <https://support.sap.com/namespaces>. To do this, your customer number must have an ABAP development license with authorization to develop your own software. You can then release the reserved 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.

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

i Note

Remember that you must [enter the namespace and define the namespace role \[page 48\]](#) after the reservation.

7.3 Add-On Software Component

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

You require 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 the Support Package Manager or the SAP Add-On Installation Tool

To ensure that the add-on software component is unique, the name of the add-on software component is automatically derived from the namespace reserved for the add-on software component. When you create an outbound delivery, the namespace prefix of the delivery name is used 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 \[page 36\]](#) 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.

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.

Example: Add-on release supports multiple SAP releases.

Your add-on supports multiple SAP releases at the same time. For example, add-on release 100 supports SAP NetWeaver 7.0 and higher. For the add-on release name, use the following syntax:

<Digit>00_<underlying SAP release>

The first digit can be freely defined, digits two and three are both 0, and the fourth place is a separator underscore. The subsequent numerals reflect the underlying SAP release. The first digit of the add-on release name and the first two places in the underlying SAP release are used for the version name of the delivery request.

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

Example: A new add-on release each year

You publish exactly one add-on release every year. For the add-on release name, use the following syntax:

<Digit>00_<year when package was created>

The first digit can be freely defined, digits two and three are both 0, and the fourth place is a separator underscore. The subsequent numerals reflect the year in which the package was created. The first digit of the add-on release name, and the first two unique places in the year when the add-on release was created, are used for the version name of the delivery request.

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

Example: Your add-on release supports exactly one SAP release.

A release of your add-on always supports exactly 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. For the add-on release name, use the following syntax:

<Digit><Digit>0

The first two digits can be freely defined and the third digit is 0. The remaining places are not used. The version name of the delivery request matches the add-on release name.

Add-on release name	Version name of the delivery request
100	100

Add-on release name	Version name of the delivery request
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 \[page 49\]](#).

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>

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 34].</p>

Part of Name	Description
<++>	<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
<namespace>	<p>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.</p>

Example

- Component list for Add-On ABCD 100_700 with reserved namespace /ABCD/:
SAPK-170COINABCD
- Support Package 03 for Add-On ABCD 200_740 with reserved namespace /ABCD/:
SAPK-27403INABCD

8 Setup of the System Landscape and Systems

To develop and maintain your add-ons, you need a specific system landscape. The structure of this landscape is described here. For information about the setup of a system landscape, see the Change and Transport System documentation:

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 the final assembly test. This test system must be updated to the same version, which is required to test a delivery, by using system copies created in advance.

The relevant consolidation system should be used as the final assembly system in all system landscapes. It should not contain any cross-client test data, in order to prevent any test data from being delivered accidentally. You can of course work with client-specific data in test clients.

i Note

In your system landscape, verify that the upgrade and maintenance strategy in the development system is compatible with the [delivery strategy \[page 29\]](#).

More Information

You can find links and paths to further information about Change and Transport System under [Further Documentation \[page 139\]](#).

8.1 System Landscape for Add-On Development

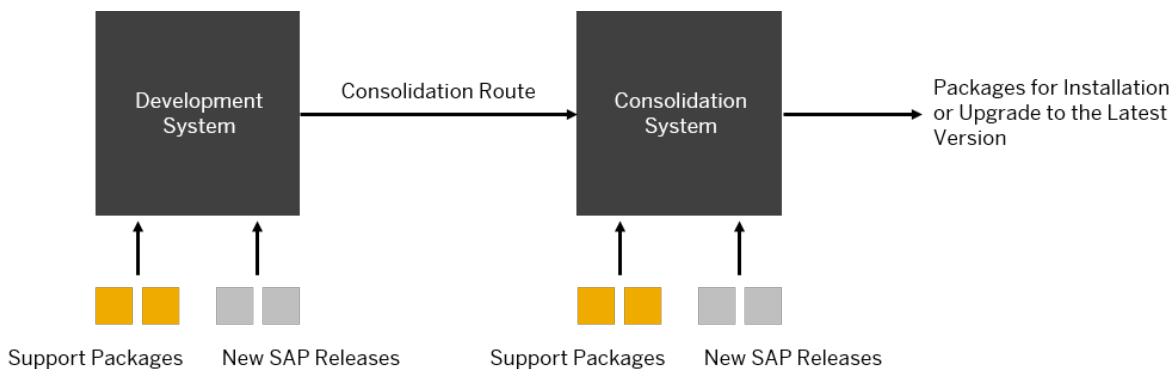
For each add-on release and each supported SAP release, you require two systems for add-on development; a development system and a consolidation system.

- **Development System:** Here you perform all activities for software and documentation development and for customizing.
- **Consolidation System:** You use this system for translation and delivery purposes. It also serves as a final assembly system.

i Note

You should **not** use this system for carrying out repairs and customizing.

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



Consolidation Route from the Development System to the Consolidation System

You also require a temporary **additional test system**. For more information, see [Test Systems \[page 54\]](#).

⚠ 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

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

1. Set up the required systems. You can do this by:

- Installing an SAP system
- Copying an installed SAP system

i Note

In both cases, when configuring your SAP system, you need to reconstruct the state you require for your [delivery strategy \[page 29\]](#). This applies in particular to the release state and support package state. When you are setting up the system landscape, also take care when creating the [client layout \[page 46\]](#).

2. Install SAP Add-On Assembly Kit in the system landscape (see [Installing SAP Add-On Assembly Kit \[page 47\]](#)).
3. Configure your system landscape for working with SAP Add-On Assembly Kit (see [Configuring the Development System and Consolidation System \[page 48\]](#)).

8.1.2 Setting Up a Development Landscape for Further SAP Releases

If you want your add-on to support multiple SAP releases, you require a separate development landscape for each release in question. This section describes the structure of the landscape for modification-free add-on development on further SAP releases.

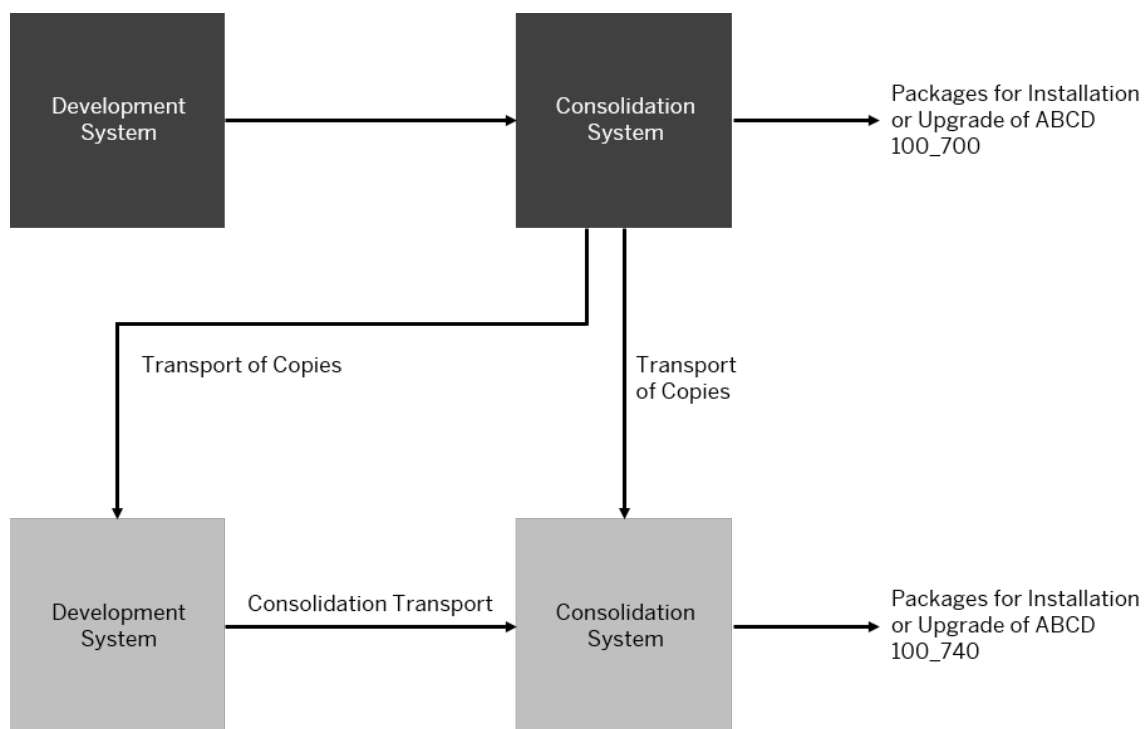
Context

The procedure for setting up a landscape for developing add-ons is described under [Setting Up a Development Landscape for Modifying Add-Ons \[page 122\]](#).

Procedure

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 \[page 47\]](#) in the systems and make the required [settings in the systems \[page 48\]](#).
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.
For more information, see *Setting Up and Coordinating Translation* under [Further Documentation \[page 139\]](#) ► [Documentation and Translation Tools](#) ►.
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 the Software Delivery Composer in the delivery client of the new consolidation system and on 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.

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.



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

Results

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

To perform maintenance tasks, you require a two-system maintenance landscape.

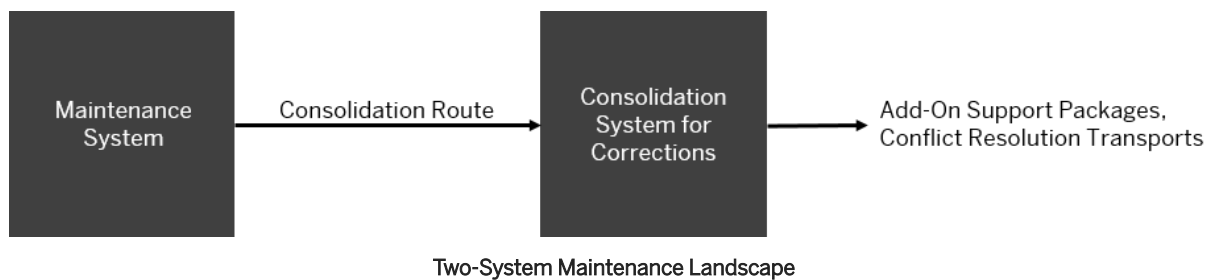
Maintenance involves the following tasks:

- Removing any errors in the add-on (which means creating add-on support packages)
- 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 \[page 124\]](#).

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



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 \[page 53\]](#)) and configure the consolidation system for corrections in the same way as the consolidation system for development.

Systems for add-on maintenance can be set up in the following ways:

- [Copy the consolidation system from the development landscape of the add-on release in question \[page 42\]](#).

→ Recommendation

We recommend the system copy as the default method.

- [Transport copies of the objects of the add-on release in question \[page 43\]](#).

8.2.1 Setting Up A Maintenance System Using a System Copy

You can set up a maintenance system by making a copy of the consolidation system in the development landscape of the add-on release.

Prerequisites

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

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.

i 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 \[page 46\]](#).)
 - 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.

8.2.2 Setting Up a Maintenance System Using a Transport of Copies

You can set up a maintenance system by using a transport of copies of objects in the add-on release in question.

Prerequisites

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

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 \[page 46\]](#)) 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 \[page 46\]](#) 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 \[page 53\]](#).
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 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 [Further Documentation \[page 139\]](#) ► [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 ensures that an add-on does not reference the objects in a different add-on. But this is only an indication and not fully reliable.
- Use the encapsulation methods available as part of the package concept. For more information, see [Further Documentation \[page 139\]](#) ► [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 information about clients and transport routes under [Further Documentation \[page 139\]](#) ► [Change and Transport System](#) ► [Change and Transport System \(Overview\)](#) ► and ► ► [Transport Management System \(BC-CTS-TMS\)](#) ►.

i Note

If not otherwise described, you always create new clients as a copy of client 000. Create the client names as specified in the pattern in the following table, replacing <x> by the digit 1 to 9.

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

- **Client Layout**

Make the following settings in table T000 (transaction SM30):

Recommended Client Layout

System	Changes and Transports for Client-Dependent Objects	Changes to Cross-Client Objects
Development System 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 <x>05)	Automatic recording of changes	No changes to repository objects
Consolidation System or Consolidation System for Corrections		
Test client (client <x>00)	Changes without automatic recording; no transports allowed	No change to repository and client-independent customizing objects
Delivery client (client <x>05)	No changes allowed	No change to repository and client-independent customizing objects

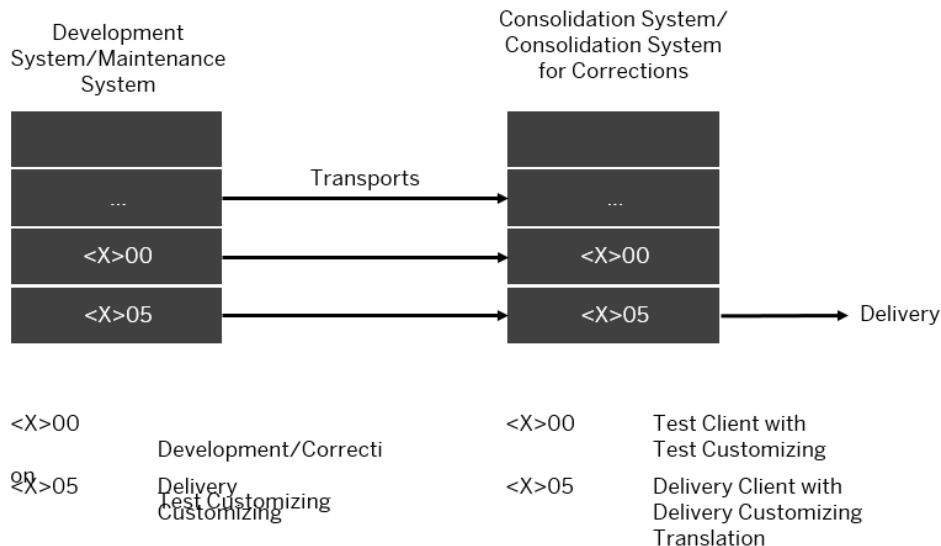
- **Transport Paths**

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

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 (<X>05) also reach test client (<X>00). For more information, see *Special Features of Extended Transport Control* under [Further Documentation \[page 139\]](#) [Change and Transport System](#).

The following figure shows how you should create transport paths.



For more information about controlling the export of language-dependent data, see [Setting Up Parameters LANGUAGE and LSM \[page 53\]](#).

For more information, see [Defining Parameters for the Transport Control Program tp \[page 52\]](#).

8.5 Installing SAP Add-On Assembly Kit

SAP delivers SAP Add-On Assembly Kit as a standalone add-on (AOFTOOLS). Use SAP 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.

Results

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

Related Information

[Configuring the Development System and Consolidation System \[page 48\]](#)

8.6 Configuring the Development System and Consolidation System

After installing the SAP Add-On Assembly Kit, you have to make a number of settings in your system landscape.

- [Entering a Namespace and Defining a Namespace Role \[page 48\]](#)
- [Creating or Updating an Add-On Software Component and Add-On Release Name \[page 49\]](#)
- [Setting a System Change Option \[page 51\]](#)
- [Configuring Parameters for the Transport Control Program tp \[page 52\]](#)
- [Setting Up Parameters LANGUAGE and LSM \[page 53\]](#)

8.6.1 Entering a Namespace and Defining a Namespace Role

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 development and consolidation systems after reserving the namespace in SAP Support Portal.

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 \[page 33\]](#).

Procedure

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 \[page 33\]](#).
- 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 \[page 33\]](#).)

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 \[page 67\]](#)). 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

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 \[page 31\]](#).

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 \[page 60\]](#), 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.

i 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

The system change option dictates whether repository objects and cross-client customizing objects are modifiable or not. You must set this option in the development and maintenance system after releasing the namespaces for your development objects.

Prerequisites

- Your user is assigned to the role SAP_AAK_SDC_CHANGE.
- You have [entered your namespace or namespaces for the development objects \[page 48\]](#).

Note

The system change option is dictated by both the namespace for the development objects **and** the software component. For more information, see *Setting the System Change Option* under [Further Documentation \[page 139\]](#) ► [Change and Transport System](#) ►.

Procedure

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 [Further Documentation \[page 139\]](#) ► [Roles and Authorizations](#) ► or choose ► [Help](#) ► [Application Help](#) ► in role administration (transaction PFCG).

8.6.4 Configuring Parameters for the Transport Control Program tp

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

Procedure

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

Parameter	Value in Development System/Main-tenance System	Value in Consolidation System/Consolidation System for Corrections
ABAP/NTFMODE	<i>b</i>	<i>b</i>
NEW_SAPNAMES	<i>true</i>	<i>true</i>
T_IMPORT	<i>no</i>	<i>no</i>
K_IMPORT	<i>no</i>	<i>yes</i>
VERS_AT_IMP	<i>yes</i>	<i>no</i>

For information about these parameters and how to configure them, see *Transport Tools (BC-CTS-TLS)* and *Changing Parameters for the Transport Control Program* under [Further Documentation \[page 139\]](#) ► [Change and Transport System](#) 🔗.

i Note

The *Transport Tool* tab in TMS shows you only the automatically generated profile parameters for the transport control program `tp`. A description of how to modify these parameters can also be found in the documentation specified above.

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

8.6.4.1 Setting Up Parameters LANGUAGE and LSM

You can control the export of language-dependent data using parameters `LANGUAGE` and `LSM` (`language_selection_mode`). Before you create and maintain add-ons, you must configure these parameters in your systems.

Context

i Note

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

Procedure

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.
This means that the language-dependent data is exported in the master language.
 - **In the Consolidation System:** Enter the value **ALL** for the `LSM` parameter.
This means that all languages that you entered for parameter `LANGUAGE` are exported. All the table entries are exported according to parameter `LANGUAGE` without you having to add the missing languages to the language-dependent tables.

i 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 Configuring the Maintenance Systems

When configuring the systems in the maintenance landscape, the same settings apply as when configuring the development and consolidation systems.

Configure the maintenance system in the same way as the development system and configure the consolidation system for corrections in the same way as the consolidation system for development.

For more information, see [Configuring the Development System and Consolidation System \[page 48\]](#).

Related Information

[System Landscape for Add-On Maintenance \[page 41\]](#)

8.8 Upgrading the Development/Maintenance Landscape

When you upgrade the system landscape in which you develop and maintain your add-ons, all objects you created yourself are preserved. These are all objects whose object directory entry specifies an original system other than SAP:

If you have made modifications to SAP objects (objects whose object directory entry specifies SAP as the original system), these modifications are overwritten by the standard SAP objects in the upgrade. You can, however, restore your modifications after the upgrade in a modification adjustment.

8.9 Test Systems

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

- **Final Assembly Test**

You use the final assembly test to test the imports of the finished packages.

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** (but recommended 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

As well as the basic rules for ABAP development, some further rules are also important when developing add-ons.

- 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.

→ Recommendation

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).

You can find basic information about ABAP development under [Further Documentation \[page 139\]](#) ► [ABAP programming basics](#) ►.

9.1 Creating and Assigning Packages

When developing your add-on, you must create and assign packages.

Prerequisites

For every add-on release, you have configured a software component in the development and consolidation system, plus a transport layer for the consolidation transports.

Procedure

1. Call transaction `SE80`.
2. Create packages for the development of your add-on.
3. Assign each package of the add-on to the associated software component and a specific transport layer,

9.2 Rules for Add-On Development

You should follow these rules when developing add-ons:

- **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 System i. Exceptions 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 `£1 BETWEEN £2 AND £3` 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 \[page 59\]](#) instead. If you want to make modifications regardless of this, follow the [rules for developing modifying add-ons \[page 125\]](#).
- **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 \[page 103\]](#).
- **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 \[page 44\]](#).
- **Client-specific objects:** Use **BC Sets** and **IMG enhancements**.
For more information, see [Further Documentation \[page 139\]](#) ► [BC Sets and IMG enhancements](#) ►.
- **Tables**
 - **Content of SAP Tables**
If you want to deliver entries in SAP tables, use BC Sets.

⚠ Caution

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.



i Note


You cannot use SAP Add-On Assembly Kit 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 on the initial screen in ABAP Dictionary (transaction SE11) in your SAP system or see [Further Documentation \[page 139\]](#)  [Delivery class of tables](#) .

- Also note the instructions under [Object List Checks \[page 76\]](#) when developing objects.
- If your add-on supports multiple SAP releases, also read [Rules for Add-On Development on Multiple SAP Releases \[page 58\]](#).
- 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 Add-On Development on Multiple SAP Releases

If you want your add-on to support multiple SAP releases, note the following rules.

- **Originality/New Objects**

If you need your add-on objects in more than one SAP Release, always create them in the lowest supported release. This has the following benefits:

- Upward compatibility is usually ensured for the objects. The ABAP syntax, for example, is usually guaranteed to remain valid in the higher release and the object types are usually known here too.
- It is usually technically possible to transport copies from a lower SAP Release to a higher SAP Release.
- If the original system of the objects is the same system, it is easier for you to keep an overview of all add-on objects.

If possible, therefore, always create new objects in the development system of the first development landscape. If you need the objects in a further development system, you can use a transport of copies to transport them to the development system in the follow-on landscape. Remember that you may need to make modifications to the objects in the higher release.

If you do not need a particular object in the first development system, you can create it in the next higher development system in the follow-on development landscape.

For organizational reasons, ensure that each object only has one original system and that in other systems the object only exists as a copy of the original.

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

Transports of copies between development systems with different releases should only be made from a lower SAP Release to a higher SAP Release. The reasons for this are discussed under *Originality/New Objects*.

- **Modifications**

If you need to make modifications in the higher SAP Release, you must implement them manually. Do not make transports of copies with modified objects, since you cannot be sure that the state of the modified objects in the higher SAP Release matches their state in the lower SAP Release.

- **Cross-Release Development**

Whenever possible, employ cross-release development to reduce development work in the various release tracks of your add-on to a minimum. See also the comments under [Defining the Delivery Strategy \[page 29\]](#).

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 under [Further Documentation \[page 139\]](#) ► [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 under [Further Documentation \[page 139\]](#) ► [Enhancement Framework](#) ►.

i Note

The information that follows focuses on individual aspects, 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 [Further Documentation \[page 139\]](#) ► [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**.

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.

i Note

This type of enhancement is not possible for pooled tables or cluster tables.

For more information about append structures in tables, see [Further Documentation \[page 139\]](#) ► [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 [Further Documentation \[page 139\]](#) ► [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.

i Note

You cannot create messages in existing SAP namespaces.

9.4 Documentation and Translation

Software development also involves writing documentation. 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).

Documentation

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

Translation

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.

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 \[page 53\]](#). 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 \[page 91\]](#).

For more information, see [Further Documentation \[page 139\]](#) > [Documentation and Translation in SAP Systems](#).

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 (AOI)
- New support package level of the add-on: Add-On Support Package (AOP)
- Adjustment with SAP support packages: Conflict Resolution Transport (CRT)
- Delta upgrade of the add-on software: Add-On Upgrade Package (AOU)
- Exchange upgrade of the add-on software (included in the SAP system upgrade): Add-On Upgrade Package (AOU) or, in exceptional cases, depending on the underlying SAP release Add-On Exchange Package (package type AOX) (see [Package Types \[page 63\]](#))

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 (up to SAP_BASIS Release 750)
- Delivery type [support package](#)
You can use the delivery order Support Package.
- Delivery type [conflict resolution transport](#)
You can use the delivery request Conflict Resolution Transport.
- Delivery Type [Pilot](#)

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:

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)

Delivery request	Package type
Exchange component piece list	Add-on exchange package (AOX, up to SAP_BASIS Release 750) or add-on upgrade package (AOU) More information about using package types can be found under Package Types [page 63] .
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:

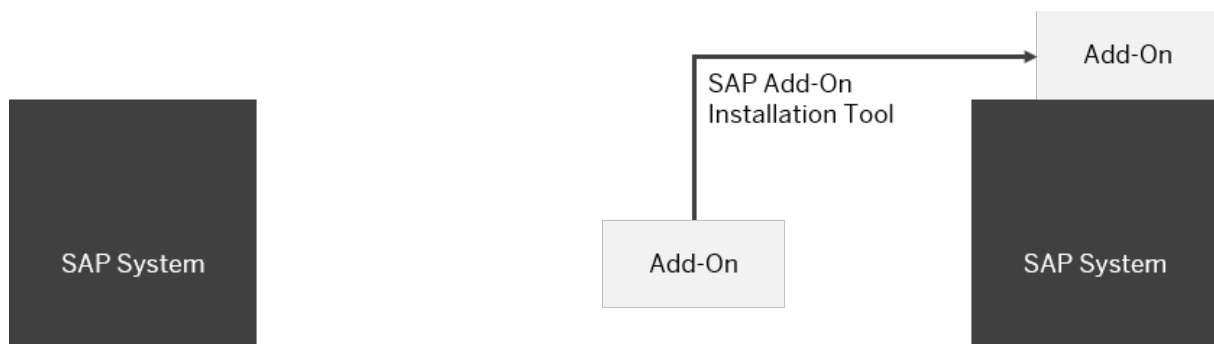
- As a compressed file via the Internet
- On a CD with a predefined directory layout (see [CD for the Add-On Delivery \[page 136\]](#))

10.1 Package Types

There are various package types available for delivering an add-on.

Add-on installation package (AOI)

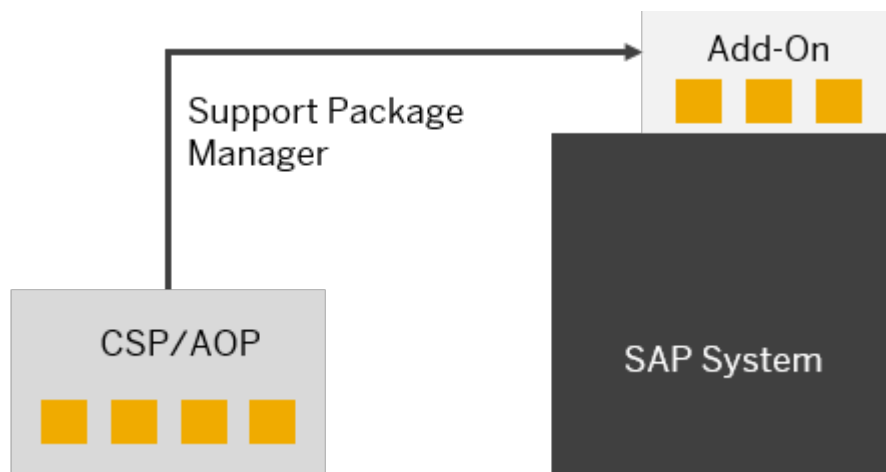
Add-on installation packages are used for the initial delivery of an add-on. It is particularly suitable for small add-ons and the extended attribute `REINSTALL_ALLOWED = T` means it can also be used for upgrades (see also [Creating Add-On Installation Packages \[page 88\]](#)). You should expect a longer downtime with larger packages. SAP Add-On Installation Tool is used to import add-on installation packages.



Initial delivery of an add-on using an AOI

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_BASIS release, the package type is either CSP (for component support packages from SAP_BASIS 6.40) or AOP (for add-on support packages up to and including SAP_BASIS 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.

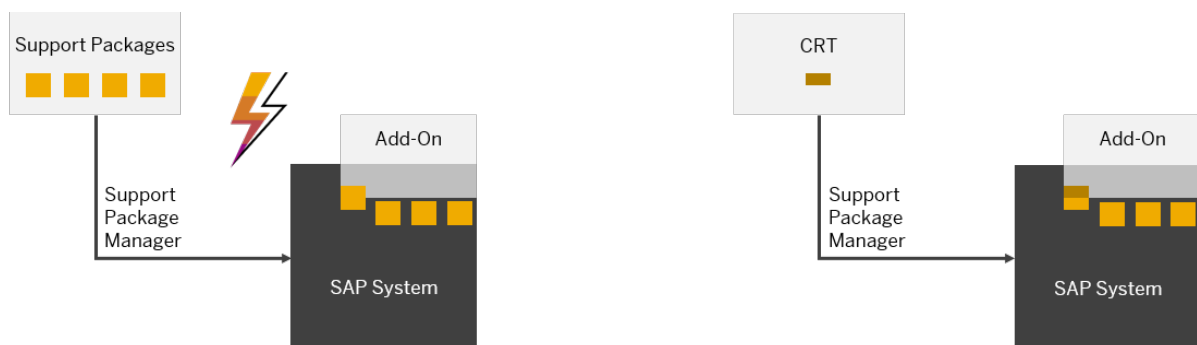


Delivery of a support package for the add-on using a CSP/AOP

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 \[page 121\]](#).

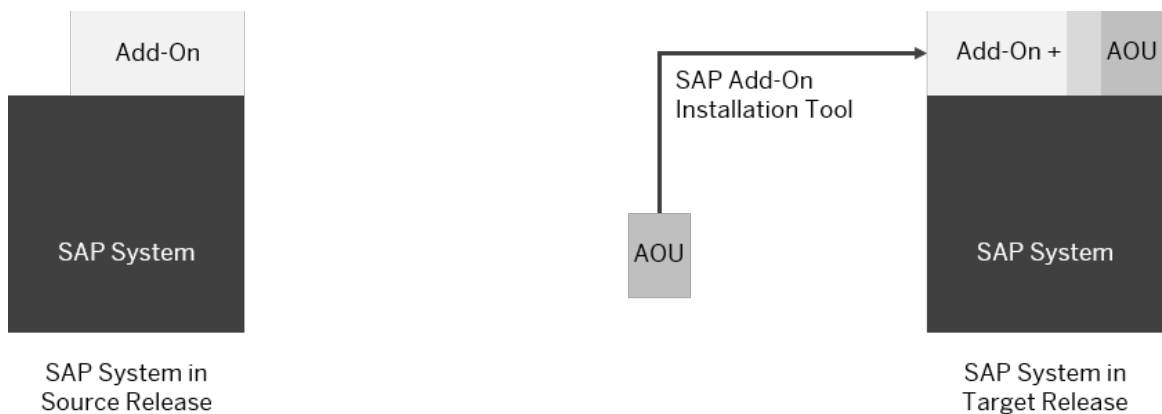


Delivery for restoring modifications using a CRT

Add-on upgrade package (AOU)

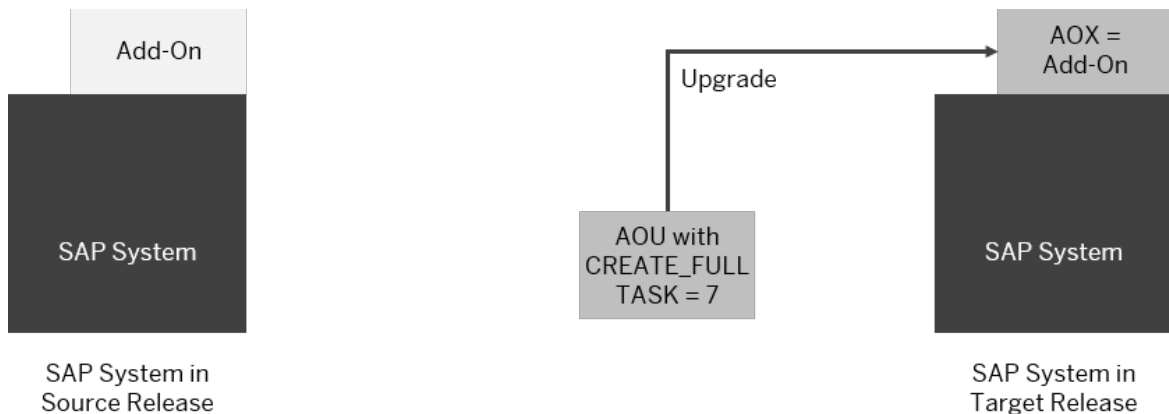
An add-on upgrade package is used to update the add-on with an unchanged SAP release (add-on delta upgrade) or for a simultaneous SAP system upgrade (add-on exchange upgrade).

- Add-on upgrade package for add-on delta upgrade
With an add-on delta upgrade the AOU's are installed using the SAP Installation Tool.



Delivery for an add-on delta upgrade using an AOU

- Add-on upgrade package for add-on exchange upgrade
These AOU's must contain the extended attribute `CREATE_FULLTASK = T` and are integrated into the upgrade during the SAP system upgrade by the SAP upgrade tool.
The table *Package Types for Add-On Exchange Upgrade* tells you when you need to use this package type.



Delivery for an add-on exchange upgrade using an AOU

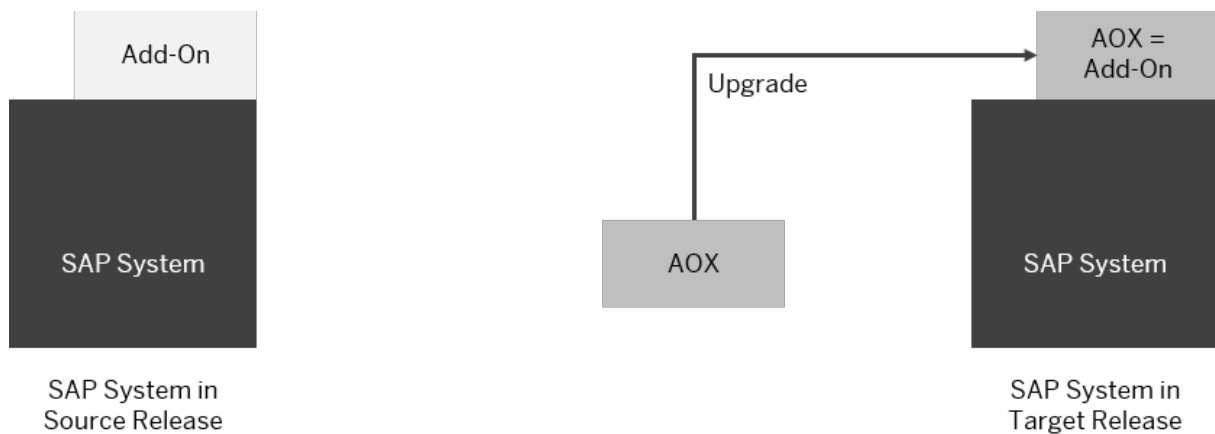
Packages of type AOU are small and only cause short downtimes because they only contain the delta (difference between start and target versions) of the add-on. A previous release must always be available for an AOU, which means that reinstallation is not possible.

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.

i Note

Add-on exchanges packages are supported but with restrictions. The table *Package Types for Add-On Exchange Upgrade* tells you when you need to use this package type.



Delivery for an add-on exchange upgrade using an AOX

Possible package types for an add-on upgrade within the SAP system upgrade

The package type, which you need to create for upgrading your add-on within an SAP system release, depends on the start release and target release of your SAP system. The following table specifies the relevant package types for all possible release combinations. The columns contain the SAP_BASIS release of the target release. The rows contain the SAP_BASIS release of the start release.

Package type for add-on exchange upgrade

SAP_BASIS	620	640	700	701	702	710	711	720	730	731	740	750	751	752
620	--	AOX	AOX	AOX	AOX	AOX	AOX	AOX	AOX	AOX	AOX	AOX	--	--
640	--	--	AOX	AOX	AOX	AOX	AOX	AOX	AOX	AOX	AOX	AOX	--	--
700	--	--	--	AOI/A OU	AOI/A OU	AOX	AOX	AOX	AOX	AOI/A OU	AOI/A OU	AOI/A OU	AOI/A OU	AOI/A OU
701	--	--	--	--	AOI/A OU	AOX	AOX	AOX	AOX	AOI/A OU	AOI/A OU	AOI/A OU	AOI/A OU	AOI/A OU
702	--	--	--	--	--	AOX	AOX	AOX	AOX	AOI/A OU	AOI/A OU	AOI/A OU	AOI/A OU	AOI/A OU
710	--	--	--	--	--	--	AOI/A OU	AOI/A OU	AOI/A OU	AOX	AOX	AOX	--	--
711	--	--	--	--	--	--	--	AOI/A OU	AOI/A OU	AOX	AOX	AOX	--	--
720	--	--	--	--	--	--	--	--	AOI/A OU	AOX	AOX	AOX	--	--
730	--	--	--	--	--	--	--	--	--	AOX	AOX	AOX	--	--
731	--	--	--	--	--	--	--	--	--	--	AOI/A OU	AOI/A OU	AOI/A OU	AOI/A OU
740	--	--	--	--	--	--	--	--	--	--	--	AOI/A OU	AOI/A OU	AOI/A OU

750	--	--	--	--	--	--	--	--	--	--	--	--	--	AOI/A OU	AOI/A OU
751	--	--	--	--	--	--	--	--	--	--	--	--	--	--	AOI/A OU
752	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

10.2 Final Assembly System

You can use your consolidation system as a final assembly system. The same settings then apply to the final assembly system as in your consolidation system.

As described in [Setup of the System Landscape and Systems \[page 38\]](#), you can use the consolidation system as the final assembly system. For more information about the required settings, see [Configuring the Development System and Consolidation System \[page 48\]](#).

i Note

Note that the same settings apply in the final assembly system for maintenance as in the consolidation system (see also [Configuring the Maintenance Systems \[page 53\]](#)).

10.3 Delivery Creation

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

The procedure that follows describes the general steps performed when creating a delivery.



- [Assembling a Delivery with SDC \[page 68\]](#)
- [Creating a Delivery Package with SDA \[page 69\]](#)
- [Testing the Delivery \[page 71\]](#)
- [Removing Errors in the Delivery \[page 72\]](#)
- [Post-Delivery of Attributes \[page 74\]](#)
- [Finishing the Creation of the Delivery \[page 75\]](#)

The way in which the delivery is created is specified by the type of the delivery and hence the package type. 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.

10.3.1 Assembling a Delivery with SDC

Use Software Delivery Composer (SDC) to assemble change requests and various administrative data for a delivery.

Prerequisites

Before you use Software Delivery Composer (SDC), import the highest available support package of SAP Add-On Assembly Kit 5.0. For this, you need the role `SAP_AAK_SDC_CHANGE`.

Context

This screen is interactive. Click the highlights for more information.



- [Creating a Delivery Package with SDA \[page 69\]](#)
- [Testing the Delivery \[page 71\]](#)
- [Removing Errors in the Delivery \[page 72\]](#)
- [Post-Delivery of Attributes \[page 74\]](#)
- [Finishing the Creation of the Delivery \[page 75\]](#)

Procedure

1. Define the type of the delivery (for example, *installation/upgrade*, *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.

When you create a delivery, the SDC, among other things, writes user IDs to a delivery log. If a delivery is deleted, the associated log containing the user IDs is deleted too.

For more information, see [Entering a Namespace and Defining a Namespace Role \[page 48\]](#)

2. 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.

i Note

You can exclude any flagged non-included change requests from the change list. They are then ignored in the delivery. To do this, choose **► Delivery Request ► Exclude Change Request ►**. You can also choose **► Change Request ► Change Request: Cancel Exclusion ►** to flag an excluded change request for inclusion in the delivery again.

3. Verify in which languages the delivery request is exported or select the languages. To do this, choose **► Delivery Component ► Select Export Languages ►**.
4. Use SDC to perform the [object list checks \[page 76\]](#). Make any corrections needed to the change list.
5. Once you have specified the content of the change list, release it in SDC.
6. 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.
7. Release the associated delivery component.

10.3.2 Creating a Delivery Package with SDA

You use the Software Delivery Assembler (SDA) to register the delivery request and create an importable delivery package from it.

Prerequisites

For this, you need the role `SAP_AAK_SDA_CHANGE`.

Context



- [Assembling a Delivery with SDC \[page 68\]](#)
- [Testing the Delivery \[page 71\]](#)

- [Removing Errors in the Delivery \[page 72\]](#)
- [Post-Delivery of Attributes \[page 74\]](#)
- [Finishing the Creation of the Delivery \[page 75\]](#)

Procedure

1. 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*.

2. Check the attributes and import conditions generated by the system. Add new attributes and conditions, change them, or delete any entries no longer relevant.

Please note the following points:

- 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 \[page 88\]](#), [Creating an Add-On Support Package \[page 97\]](#), [Creating a Conflict Resolution Transport \[page 126\]](#), [Creating an Add-On Upgrade Package \[page 98\]](#), [Add-On Behavior in SAP System Upgrades \[page 99\]](#)) 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.

3. Register the delivery request.

This provides you with an importable package with the status *Locked*.

i Note

If a package has already been registered in another system, you can migrate it to your current system. For more information, see [Migrating Packages \[page 86\]](#).

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.

4. 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*.

10.3.3 Testing the Delivery

Before you can release a delivery package, test it in a test system for completeness and correctness.

Context



- [Assembling a Delivery with SDC \[page 68\]](#)

- [Creating a Delivery Package with SDA \[page 69\]](#)
- [Removing Errors in the Delivery \[page 72\]](#)
- [Post-Delivery of Attributes \[page 74\]](#)
- [Finishing the Creation of the Delivery \[page 75\]](#)

Procedure

1. 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`.

2. 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.

3. Then import the package in your test system.

Use the appropriate import tool to do this.

4. Test the functions in the package.

10.3.4 Removing Errors in the Delivery

If errors occur during the testing of the delivery, you must correct them before releasing the delivery package. The procedure for this depends on the type of error.

Context



- [Assembling a Delivery with SDC \[page 68\]](#)
- [Creating a Delivery Package with SDA \[page 69\]](#)

- [Testing the Delivery \[page 71\]](#)
- [Post-Delivery of Attributes \[page 74\]](#)
- [Finishing the Creation of the Delivery \[page 75\]](#)

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

Procedure

- Correct incorrect or incomplete **attributes or import conditions** in Software Delivery Assembler (SDA) before you register the package again. Proceed as follows:
 - a. In SDA, go to the tab for your package type and enter the name of the package.
 - b. Choose [Change Attributes](#) in the registration options.
 - c. Correct the errors in the attributes and import conditions.
 - d. Register the package again.
 - e. Test the delivery package.
- In the case of incorrect or incomplete **contents of the delivery request**, you cancel the release of the delivery component in Software Delivery Composer (SDC). 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:
 - a. Reset the delivery component in Software Delivery Composer.
 - b. Correct and check the delivery request.
 - c. Release the delivery request again.
 - d. Release the delivery component again.
 - e. 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](#).
 - f. Test the delivery.

10.3.5 Post-Delivery of Attributes

It is often necessary to add new import attributes or correct existing attributes after the delivery of OCS packages (such as add-on installations, 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.

Context



- [Assembling a Delivery with SDC \[page 68\]](#)
- [Creating a Delivery Package with SDA \[page 69\]](#)
- [Testing the Delivery \[page 71\]](#)
- [Removing Errors in the Delivery \[page 72\]](#)
- [Finishing the Creation of the Delivery \[page 75\]](#)

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:

Procedure

1. Go to the tab of the package type for which you want to post-deliver attributes.
2. Enter the package name and choose [Select Registration Option](#).
3. Choose [Post-Delivery](#) and confirm by choosing [Continue](#).
4. 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.
5. Confirm your changes by choosing [Continue](#).
 6. Choose [Register](#).
 7. Confirm that they are correct by choosing [Yes](#).
 8. Confirm the EPS file by choosing [Continue](#).
 9. Release the ACP either as a standalone package or together with another package (with a package type other than ACP). This is known as a carrier 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 the ACP is released without a carrier package, delivery takes place automatically with the next package released, which then becomes the carrier package. Both packages (the ACP and the carrier package) must be made available for the import.
 - 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. The full procedure is illustrated in the following video:

i Note

If you want to view which attributes are delivered as post-delivery attributes in an existing ACP, you can use the package for which the ACP was created. To do this, first display the package attributes for the package in SDA and then open the attribute editor for post-deliveries under [Attribute Post-Delivery](#). You now see the extended attributes and the import conditions assigned to the package containing the delivery of the ACP.

10.3.6 Finishing the Creation of the Delivery

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

Context



- [Assembling a Delivery with SDC \[page 68\]](#)
- [Creating a Delivery Package with SDA \[page 69\]](#)
- [Testing the Delivery \[page 71\]](#)

- [Removing Errors in the Delivery \[page 72\]](#)
- [Post-Delivery of Attributes \[page 74\]](#)

Procedure

1. Confirm the delivery in SDC.

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

2. 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.

10.3.7 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 uses the object list checks to support you when searching for objects that cannot be delivered.

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 [My Object List Checks \[page 80\]](#).


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. More information can be found in 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.

i 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 deliver 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.

Caution

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

You are also not allowed to modify more than one 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.

Table entries in the customer range (protected in accordance with TRES C)

Table entries and logical entries cannot be in ranges reserved for customers (in accordance with table TRES C).

Add-on uninstallation: Non-deletable objects

Add-on uninstallation 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 such objects have been delivered, the add-on can only be uninstalled if a plug-in class is defined for this add-on. In this plug-in class, you use method `GET_OBJECT_CLASSIFICATIONS` to define how critical objects are to be handled during deletion. For more information, see [Plug-In Interface for Add-Ons \[page 113\]](#).

This method implementation can be taken into account during execution of the check if the name of the plug-in class is stored as a parameter for the specific delivery, check *Add-on uninstallation: Non-deletable objects* (CL_DL V_OLC_ADDON_DEL_CHECK), and attribute name `DEINSTALL_PLUGIN`.

For more information about parameters for object list checks, see [Specifying Parameters for Object List Checks \[page 85\]](#).

Generic checks and automatic corrections

The content of the delivery request is checked for obsolete and out-of-date objects. Any changes needed to the current object types (for example, replacements by logical transport objects) are made automatically. These checks are always performed.

You can modify the behavior of this object list check manually. For more information, see [Configuring Automatic Checks and Corrections \[page 85\]](#).

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 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.

i 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

Consistent objects

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*

Full objects in delivery requests

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.

10.3.7.1 My Object List Checks

If you need further object list checks for delivery requests in addition to those delivered with the standard system, you can specify these 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.

i 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](#).

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_EM_OLC~CHECK`. Here, access to the data of the request in question is provided by the import 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 `PGMID`, `OBJECT`, and `OBJ_NAME` (and `LANG` and `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:

- **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.

Use of the Parameters

In the source code of the check, you can access the parameter values (see also: [Specifying Parameters for Object List Checks \[page 85\]](#)) using a SELECT to database table TRDELVCHKA:

Sample Code

```
SELECT SINGLE value FROM trdelvchka INTO lv_value
WHERE dlv_name    = <name of delivery>
  AND check_name = <name of check>
  AND attribute   = <name of attributes>.
```

In the context of a class-based check, you can determine the name of the delivery using a SELECT to the delivery request table:

Sample Code

```
SELECT SINGLE * FROM trdelvtask INTO ls_trdelvtask WHERE dlv_trkorr =
g_request-h-trkorr.
```

The field `g_request-h-trkorr` is then provided by the interface `IF_EM_OLC` and filled by the framework of the object list checks. The value in field `ls_trdelvtask-dlv_name` is the name of the delivery.

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.

i 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.

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.

10.3.7.1.1 Adding Your Own Object List Checks and Documentation

In Software Delivery Assembler, you can add your own object list checks that apply when creating deliveries and document any manual changes made to the check results.

10.3.7.1.1.1 Activating Additional Checks

To include your own object list checks when creating the delivery, you have to activate them.

Context

Note the following:

- The order in which the object list checks are performed can vary from system to system. It is not defined.
- 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 different object sets in a request.
 - Objects
 - Table keys
 - String-like table keys

Procedure


1. In Software Delivery Composer, choose ► *Utilities* ► *Add Object List Checks* ► for a delivery and confirm the dialog field.
2. On the next screen, choose *New Entries*.
3. 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 the 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, see [Further Documentation \[page 139\]: ► Documentation and Translation in SAP Systems](#) .

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

You can accept negative check results of the object list check manually, but you must document the reason for this procedure.

Context

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 your own text for this or amend or add to predefined text. A predefined text is provided as a default. You can define further texts.

Procedure

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.

10.3.7.2 Specifying Parameters for Object List Checks

The specification of additional parameters and attributes can influence the function of object list checks, specifically for each check and delivery. This can be necessary both for user-developed and SAP-delivered object list checks if, for example, an add-on is to be uninstalled containing non-deletable objects.

Prerequisites

If you create your own object list checks, it may be necessary to store and define your own parameters for these checks which can provide values for a specific check module. To do this, call Customizing table **TRDELVCHKATTR** in transaction **SE16**.

The entries required for the checks delivered by SAP are included in the delivery and do not require manual changes.

Procedure

1. In the Software Delivery Composer (transaction code **SSDC**), open the required delivery, select [Object List Check](#) and then [OLC Parameterization](#) (**Ctrl** + **F8**).

Alternatively, in table view maintenance (transaction code **SM30**), you can also enter table names **TRDELVCHKA** and choose [Maintain](#).

The existing parameters for all deliveries will be displayed.

2. To create a new entry, choose [New Entries](#), enter the necessary information and save the new entry.

i Note

For check Add-on uninstallation: Non-deletable objects **CL_DLV_OLC_ADDON_DEL_CHECK**, attribute name **DEINSTALL_PLUGIN** is defined. In this case, enter the name of the plug-in class as the [Attribute Value](#) (see also [Plug-In Interface for Add-Ons \[page 113\]](#)).

The **+** field is a numerator. If only one attribute is specified, this field can remain empty. If the same attribute is specified more than once, you must assign different numeric values (for example 0, 1, 2, 3) to differentiate between the individual entries.

10.3.7.3 Configuring Automatic Checks and Corrections

The object list check **Generic Checks and Automatic Corrections** is applied automatically when you include requests in a delivery request. The proposed corrections are always applied. This modifies the content

of the checked piece list automatically. You can configure the behavior of this object list check by adding new checks or removing existing checks.

Procedure

- **Removing Automatic Checks**

In this case, you no longer want a check to run automatically when a change request is included in a delivery request.

→ Recommendation

We recommend that you perform any checks that you remove from the list manually.

- a. In transaction `SE16`, display the table `EMCHECK` and choose *Execute*.
- b. Select the check class that you want to remove from the object list check (for example check class `CL_EM_OLC_DLVUNIT_HOME`), delete the entry in the `AAK` column, and save your changes
The check now no longer runs automatically.
- c. To apply the check manually, display your delivery request in Software Delivery Composer (transaction `SSDC`), choose **Utilities** > *Add Object List Check*, and select the check in question.
The check is now in the list of manual checks.

- **Adding Automatic Checks**

In this case, you want a new check to run automatically when a change request is included in a delivery request.

- a. In transaction `SE16`, display the table `EMCHECK` and choose *Execute*.
- b. Create a new entry with the appropriate check class and enter the value `x` in the field `AAK`.

The change now runs automatically.

10.3.8 Migrating Packages

If you have registered a package in a different system, you have the option of migrating it out of that system and into your current system. Here, all information is taken from the package. The package is then available for delivery, in the same way as it is after registration.

Context

i Note

In this way you can also prepare language packages for add-on installations and add-on upgrades for delivery. The following prerequisites apply to migrations of language packages:

- You have **not** used the language export wizard to perform the export.

- You have exported the language for one add-on only.
- The package has an EPS file name that follows the naming convention: <SID><installation number>_<7 figure number>.PAT

Procedure

1. To migrate a package, choose ► [Extras](#) ► [Migrate Package](#) ►.
2. Under [Package Migration](#), choose one of the following options:

- **Package (SSDA)**
Select this option if you used SDA to register the package in another system for which an RFC connection exists.

→ Recommendation

Whenever possible, always use this option to migrate a package. Using this option enables administration information, such as the user who created the package, to be applied. This information is lost when the migration is performed using the EPS file. The type of migration is ignored when importing packages with Support Package Manager or SAP Add-On Installation Tool.

Enter the name of the registered package.

- **EPS file (EPS/out)**
Select this option if the package is registered in a different system for which no RFC connection can be established. A prerequisite for this is that the associated EPS file of the package is located in the transport subdirectory `EPS/out` of a system for which an RFC connection is possible. This can also be the current system.

i Note

You must always use this option to migrate language packages. You cannot use the SDA to register language packages directly.

Enter the name of the EPS file.

3. Enter the RFC destination for the system from which you want to migrate the EPS file of the package.
4. Choose [Continue](#).

Results

The package was migrated and has the status [Locked](#). The package was replicated to the follow-on systems with a test signature. You can now request the package for a test system.

10.4 Creating Add-On Installation Packages

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).

Context

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 \[page 98\]](#) 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 \[page 67\]](#)).

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 Providing a Delivery

If you have created the add-on package with the SAP Add-On Assembly Kit, then your add-on will be delivered with all necessary information. Please note the following points:

Procedure

- Decide how you want to deliver your add-on package:
 - Delivery on CD
 - Delivery in compressed format via the Internet
- Ensure that your delivery package contains the add-on software and the required documentation, including the installation guide (see: [Template: Installation Guide \[page 89\]](#)).

- Ensure that your CD has the correct structure and the delivery data is included as a SAR archive in the CD subdirectory DATA (see: [CD for Add-On Delivery \[page 136\]](#)).

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.

i 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

1. Prerequisites for the installation of <add-on><release>
2. Preparation of the installation of <add-on><release>
3. Installation of <add-on><release>
4. Follow-up of the installation <add-on><release>
5. Password for the installation of <add-on><release>
6. Language support

1. 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:

<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: Installed Software Component Versions](#) ►):

❖ Example

SAP_BASIS	740
SAP_BW	740
PI_BASIS	740
or PI_BASIS	730

- Prerequisite support packages:

❖ Example

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

- Additional information about the installation

❖ Example

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

2. Preparing 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 SAP Add-On Installation Tool. To do this, choose the help function on the toolbar and then choose ► [Online Documentation](#) ► [Loading Installation Packages](#) ►.

3. 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 use 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 SAP Add-On Installation Tool. To do this, use the help function in the toolbar.

4. After the Installation of <add-on><release>

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

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

6. 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

It is possible that you may need deliver a language for an add-on that has already been delivered. To do this, you can create an add-on language package created or use a subsequent support package.

We recommend that your initial add-on delivery contains all languages your require. This means that all languages must be fully [translated \[page 60\]](#). If you have configured the [parameters for exporting language-specific data \[page 53\]](#) 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.

❖ Example

You use your add-on in your company and your company merges with a company in a foreign country with a national language that has not been translated for the add-on. The functions of the add-on are available, but the new company needs the add-on texts in a different language. Or you sell your add-on to a foreign company, which is interested in having additional languages.




You can deliver languages retroactively in the following ways:

- [Deliveries Using an Add-On Language Package that Only Contains the Translation \[page 92\]](#)
- SAP NetWeaver 7.00 and higher: [Deliveries Using a Subsequent Support Package \[page 94\]](#)

10.6.1 Delivering Languages Using a Add-On Language Package



If you want to deliver subsequently translated languages using an add-on package, proceed as follows:

Prerequisites

- You have already delivered your add-on without the new target language. You may have also 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 using [SAP Globalization Services](#) .
- 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 imports, see *Language Transport* under [Further Documentation \[page 139\]](#):  [Change and Transport System](#) .



Procedure



1. Prepare the consolidation system for corrections for the translation of the add-on into the new target language

For more information, see *Setting Up and Coordinating Translation (BC-DOC-TTL)* under [Further Documentation \[page 139\]](#):  [Documentation and Translation in SAP Systems](#) .

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. Ensure that the previously determined objects are translated into the new target language.
For more information, see *Tools for Translators (BC-DOC-TTL)* under [Further Documentation \[page 139\]](#):  [Documentation and Translation in SAP Systems](#) .
3. Create an add-on language package for the new target language using the language export tool (transaction SMLT_EX).

For information about language exports, see *Language Transport* under [Further Documentation \[page 139\]](#):  [Change and Transport System](#) .

Note the following points when creating the language package using the language export tool:

- 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?](#). Confirm.

Caution

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. Make sure that the language package is available in transport subdirectory `EPS/out/` on the consolidation system after the export is complete.


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.
- a. To do this, copy the language package from transport subdirectory `EPS/out` on the consolidation system to the transport directory `EPS/in` on the test system for corrections.
 - b. 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.
 - c. 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](#) .
 - d. Use the language import tool (transaction `SMLT`) to import the language package in the test system.
 - e. Check that the full add-on is available in the new target language.
6. Migrate the language package to Software Delivery Assembler in the system where you registered your other packages.

This ensures that all packages your 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. If you have successfully completed the tests, you can [provide the language package for delivery \[page 88\]](#).

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. Add the new target language to your consolidation system for corrections in [Parameter LANGUAGE \[page 53\]](#).

This is necessary because from now on, you will also need to maintain the add-on in the new target language. 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.

i 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:

1. Install the add-on package and all support packages created before the language package was created
2. Import the language package
3. Import any further support packages that already contain the new language package

10.6.2 Delivering Languages Using a Support Package

If you want to deliver subsequently translated languages using a support package, proceed as follows:

Procedure

1. Create an up-to-date 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, see *Tools for Translators (BC-DOC-TTL)* under [Further Documentation \[page 139\]](#):

► [Documentation and Translation in SAP Systems](#) ►.

3. Use the language export tool (transaction `SMLT_EX`) to export the new languages (as described in [Delivering Languages Using a Add-On Language Package \[page 92\]](#)).
4. Use Software Delivery Composer to include the new export in the new support package of the add-on. You must explicitly select the new translated target languages 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.

This registers the new delivered languages as “installed” when importing to a target system. The languages specified in the mandatory attribute `LANGUAGE` must exist and must 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 more information, see the *simplification database* under: ► [Custom Code Migration](#) 📄

If the add-on can be used, but further preparations are necessary during the switch, SAP provides a framework. Here you can provide functions for prechecks and define the compatibility between SAP S/4HANA and the various software components in accordance with 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 \[page 76\]](#).
- If your add-on contains modifications to a different software component, note also the [rules for developing a modifying add-on \[page 125\]](#).

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 \[page 97\]](#)).
- In **modifying** add-ons:
Creating conflict resolution transports (CRTs) (see [Creating Conflict Resolution Transports \[page 126\]](#)).
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

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

Context

Note also the restrictions that apply to permitted changes. You can find these in [Rules for Add-On Maintenance \[page 97\]](#).

Procedure

1. Make the corrections in the maintenance system.
2. Import the corrections into the consolidation/final assembly system for corrections.
3. Create the delivery in the delivery client of the consolidation system for corrections by using the tools in the SAP Add-On Assembly Kit (see [Creating Deliveries \[page 67\]](#)).

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)

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).

Context

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

Please note that you need to create different package types (depending on the start release and target release), in order to update add-ons during an SAP system upgrade. The table *Package Types for Add-On Exchange Upgrade* (under [Package Types \[page 63\]](#)) tells you when you need to use which package type. The procedure is described under [Updating Add-Ons in SAP System Upgrades \[page 100\]](#).

Procedure

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 with the tools of the SAP Add-On Assembly Kit (see [Creating Deliveries \[page 67\]](#)).

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

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

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 \[page 54\]](#).

You have the following options:

- You can update your add-on during the SAP system upgrade.

→ Recommendation

We recommend this option as the default method.

In this case, you need to create a new release of your add-on. Using the SAP Add-On Assembly Kit, you can define different package categories (depending on the source and target release of your SAP system) to ensure the package is included at the correct place in the upgrade. For more information about this procedure, see [Updating Add-Ons in SAP System Upgrades \[page 100\]](#).

- 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 \[page 102\]](#).

- 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.

11.4.1 Updating Add-Ons in SAP System Upgrades

If your upgrade strategy involves updating the add-on during the SAP system upgrade, then proceed as described here.

Procedure

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 with the tools of the SAP Add-On Assembly Kit (see [Creating Deliveries \[page 67\]](#)).

Please note the following points:

1. In Software Delivery Composer, use the delivery type *Installation/Upgrade* to create a delivery request of the type Exchange Component Piece.

i Note

If you use the software component SAP_BASIS in Release 751 or higher, it is no longer necessary to create an exchange component piece list. Create a change piece list or component piece list instead.

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 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:
 - Tab page AOU: Creates an add-on upgrade package
This action sets the extended attribute `CREATE_FULLTASK` to the value `T`. This ensures that the AOU is accepted as an upgrade package during the SAP system upgrade.
 - Tab page AOX: Creates an add-on exchange package

i Note

The package type AOX is only partially supported. To find out when you need to use which package type, please refer to [Package Types \[page 63\]](#).

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 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 \[page 136\]](#) 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 \[page 136\]](#).

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 \[page 88\]](#)).

Results

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 relevant release under <https://support.sap.com/sltoolset>  [System Maintenance](#) .

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.

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 \[page 99\]](#).

Related Information

[Add-On Uninstallation \[page 103\]](#)

12 Add-On Uninstallation

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

Technical Requirements

- The system is based on SAP NetWeaver 7.0 or higher.
- You have installed at least SPAM/SAINT Version 0053 or the version used to test the deletion.

→ Recommendation

We recommend that you always import the newest version of a SPAM/SAINT update. The current version is available on SAP Support Portal, under <https://support.sap.com/swdc>.

- You are using a kernel with at least Release 7.2.
- You are using the transport tool tp in Version 380.07.22 or higher.
- The transport tool R3trans has at least the version from August 6, 2013.
- The final ACP of the add-on you are uninstalling is installed in the system

Uninstallability of Add-Ons

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.

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:

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 add-on that can be uninstalled:

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.

Activity	Procedure
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	Get an overview of which objects and names are used that were created by the add-on.
Check objects created by the add-on	To identify add-on-specific customizing settings, you must follow naming conventions in the development phase.
Identify generated objects in connected systems	Verify whether there are any generated objects that might no longer exist if the add-on is uninstalled.
Check data in tables outside of the add-on	If you need to modify the add-on, check whether it uses any tables that are part of another add-on.
Check reimports of the deleted add-on	Check what happens if a customer installs a deleted add-on again.
Remove customizing settings	Any data inserted in tables manually must be removed before the add-on is deleted.
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 if the ad-on was successfully deleted.	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:

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.

Activity	Procedure
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> i Note The add-on cannot be deleted if it is not possible to access data archives. </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:

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.

Activity	Procedure
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> i Note Once the add-on is deleted then you can no longer complete the workflows. </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> i Note An add-on can only be deleted when it is no longer in use. </div>
Check Dependencies Between Other Systems and the Add-On	<p>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.</p>
Adjust Central Monitoring	<p>If the add-on is connected to the central monitoring by providing data for this? If yes, then this must be set accordingly.</p>
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 uninstallable add-on:

Activity	Procedure
Validate the uninstallation process	<p>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.</p>
Test whether the deletion was successful.	<p>Ensure that you know what you need to test in order to determine if the add-on deletion was successful.</p>

Activity	Procedure
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 uninstallable if it has passed the following check steps.

Procedure

1. Set up a test system.

To test the uninstallation, use a suitable test system. If necessary, set up a new system for this purpose. 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. Generate realistic application data so that runtime data can be taken into consideration as well in the uninstallation.

2. 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>.

3. Simulate the uninstallation of the add-on and process all critical objects in the test system.

For more information, see [Simulating an Add-On Uninstallation \[page 110\]](#) and the online documentation of the tool. To do this, use the help function in the toolbar.

4. 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).

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

For the system to recognize the add-on as uninstallable, the attributes `DEINSTALL_ALLOWED` and, if required, `DEINSTALL_PLUGIN` must be set for the installation package. If you want to make an add-on uninstallable 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 uninstallable 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, see [Post-Delivery of Attributes \[page 74\]](#).

6. Then import the ACP into your test system.
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 plan⁴ [page 108] 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 uninstalleable.

→ 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 uninstalleable.

Inform your customers about the uninstalleability 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 uninstalleable. 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 Simulating an Add-On Uninstallation

Simulate the uninstallation of your add-on to ensure that it can be deleted.

Prerequisites

- For the simulation, you use a system in which the add-on to simulate is installed using a standard installation (with the tools SPAM/SAINT). From SPAM version 78, you can also use the final assembly system in which the add-on is available by SDC delivery.
- The system in which the simulation is performed is based on SAP NetWeaver 7.0 or higher.
- You have imported at least SPAM version 74.

i Note

You cannot delete add-ons with the simulation tool.



Procedure

1. Call the simulation tool in your system using transaction code **SIM_UNINST**.

The initial screen shows you a list of all software component versions that are installed in or exist in your system. For each entry, you also see information about whether a simulation was already made for this software component version in the past and whether it can be uninstalled.

2. Select the software component version you want to uninstall and start the simulation by choosing *Simulate*.

i Note

If there is already a result from a previous simulation ( or  in the *Simulation* column), you must decide before a new simulation whether to resume the simulation with the existing result or whether you want to start again.

Once started, the results of a simulation are preserved even if you reset the simulation.

If a simulation for a software component version has been completed, it can nevertheless be useful to execute the simulation again. This is always the case if the initial situation changes in the system where the simulation is running, for example when you import a new support package.

i Note

If a simulation is not yet complete, the simulation queue is active and the add-on is locked for all Lifecycle Management actions in this system. When attempting to repeat a simulation for the same software component version, the *Simulation Results* table is displayed directly.

The status line of the tool enables you to monitor the progress of the simulation.

After the simulation, the [Simulation Results](#) table shows all objects that would produce errors in an uninstallation. You need to decide whether to ignore these objects or delete them. These critical objects are highlighted in yellow or red. The [Comment](#) field tells you why the object is regarded as critical (see: [Critical Objects \[page 112\]](#)). You can also edit this field, for example to document your decision making. The [Proposal](#) column proposes a method for handling the simulation tool for each object. For more details, click the proposal in question.

If you want to modify the display of the [simulation results](#) table, you can use predefined filters or your own filters. For more information, use the help button in your system.

If, after a simulation run, the full list of all objects that belong to the software component version in question is displayed without red and yellow entries, the simulation tool was able to define an operation for all objects. This means that there are no critical objects for which you have to make your own decisions.

3. To enter your operations for critical objects in the [Simulation Results](#) table, switch to change mode.
4. For each object, define how you want it to be handled in the uninstallation. The following options are available:
 - Edit each object individually by selecting [Delete](#) or [Ignore](#) in the [Operation](#) column.
 - Apply all operations proposed by the simulation tool at once by choosing [Apply Proposals](#).

i Note

If multiple objects with different proposed operations exist for an object type, this is shown under [See Details](#). In this case, check and apply the proposed operations individually in the [Simulation Results](#) table.

Double-clicking an entry displays further information about the entry by object type in an additional [Simulation Results – Details](#) table.

→ Recommendation

For the following entries, always check the [Simulation Results – Details](#) table and edit the individual items displayed in it if necessary:

- Entries in the [Simulation Results](#) table that have * as the object name. This indicates that multiple objects exist for this object type. Double-click the entry to display all related objects.
- Entries of the object types TABU, VDAT, and CDAT

If multiple objects exist for an entry in the [Simulation Results](#) table, you can define the same operation for all related objects centrally. To do this, edit the entry in the [Simulation Results](#) table. If you want to handle the related objects in different ways, choose the appropriate operations for the individual entries in the [Simulation Results – Details](#) table. The [Operation](#) column in the [Simulation Results](#) table shows this in the [See Details](#) entry.

5. Once you have defined an operation for all critical entries, choose [Simulate](#) again.
The status line of the tool enables you to monitor the progress of the simulation.
6. When the simulation is complete, check the results. To do this, choose [Summary](#).

→ Recommendation

Read this summary even if the simulation does not find any critical objects.

You are given an overview of the results of the simulation and a list of the activities you must perform to make the add-on in question uninstalleable.

If you need to create a plug-in class, the simulation tool generates code from the simulation information and you can use this code directly. The decisions made in the simulation are applied automatically to the actual uninstallation.

Choose [Print/Export](#) to transport or print this summary in an appropriate format. The results of the [Simulation Results](#) table can be saved to a spreadsheet for further processing ([Export to Spreadsheet](#)).

12.6.1 Critical Objects

Critical objects cannot be processed during an uninstallation and hence cancel the deletion.

In a simulation, the simulation tool checks how all objects of the software component version in question are handled during the deletion. To do this, the tool uses relevant information from the system and (if available) the plug-in class for an uninstalleable add-on in which the required decisions are already implemented. Objects for which the simulation tool cannot define an operation are referred to as critical objects. They would cause the deletion to be canceled during a real uninstallation. For these objects, you must first decide how you want them to be handled during the uninstallation.


The simulation tool distinguishes the following critical objects in the [Simulation Results](#) table:

- Red entry: Decision required.
- Yellow entry: Decision for current situation in the test system is not mandatory. If the test system does not reflect the identical state of the customer system, a real uninstallation can still be canceled.

For each critical object, the [Simulation Results](#) table displays a comment that describes the error and supports you in the decision making process.

12.7 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 uninstalleable. 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 SAP 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 \[page 113\]](#).

For examples of how to set attributes, see [Examples: Attributes for Add-On Uninstallations \[page 134\]](#).

12.8 Plug-In Interface for Add-Ons

An add-on does not generally have to be changed to be deletable. However, uninstallation behavior can be influenced by add-on-specific data and checks.

An add-on can support its uninstallation using its own methods:

- You can define a check that you use to check if an add-on can be deleted before the deletion.
- You can define a check that you use to check, before the deletion, if dynamically generated objects are to be included in the deletion.
- You can define a check that you use to decide whether the add-on data is to be retained or deleted.

Note

Note that data should be retained in justified exceptional cases only and, as a rule, all data should be deleted if possible.

Caution

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

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 plug-in class to be implemented. If this attribute is set, the uninstallation checks whether 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 attribute `DEINSTALL_PLUGIN`.

If the instantiation was successful, the system checks whether the interface methods exist and calls them if they do. (At least the 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 whether the add-on can be deleted. It calls the 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 the method `GET_CREATED_OBJECTS` and puts all objects in to the deletion task. The method `GET_OBJECT_CLASSIFICATIONS` is used to define the deletion behavior of the objects and table entries.

Note

Add-on plug-in class `CL_OCS_PLUGIN_TEMPLATE`, which you can use as a template for creating your own class, is delivered with SPAM update 63.

Methods of Add-On Plug-In Class

The implementation can only work if an interface has certain properties, listed in the following.

i Note

The export parameters `EV_RC` and `ET_MESSAGES` are relevant for all methods. The following applies to them:

- `EV_RC`: This parameter indicates whether the method was called correctly and was completed successfully or not. It does not provide any information about the content of the result of the method in question. Possible values:
 - `EV_RC = 0`: The method call was technically correct and successful.
 - `0 < EV_RC <= 4`: The method call ended with warnings but the uninstallation can be resumed anyway.
 - `EV_RC > 4`: The method call ended with errors and the uninstallation was canceled.
- `ET_MESSAGES`: This parameter is used in failed or very complex checks of to provide messages useful for troubleshooting. If the method call is successful (`EV_RC = 0`), the table can remain empty.

- Static factory method `GET_REFERENCE`

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:

- Import parameters:
 - `IV_COMPONENT - TYPE DLVUNIT`: Name of the add-on to uninstall
 - `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 of log entries (can be empty)
 - `EV_RC - TYPE INT4`: Specifies whether the method call was successful.
- Instance method `CHECK_PRECONDITIONS`

You can use this method to check whether all prerequisites required for the uninstallation of the add-on are met. For example, this method verifies whether all prework was completed or whether a function was only used in a specific mode. The result of this check is specified by the export parameter `EV_CHECK_RESULT`. 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 of messages (can be empty)
 - `EV_RC - TYPE INT4`: Specifies whether the method call was successful.
- Instance method `GET_CREATED_OBJECTS`

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 of transport objects to delete
 - `ET_E071K - TYPE TR KEYS`: Table of table keys (as required)
 - `ET_MESSAGES - TYPE SPROT_U_TAB`: Table of messages (can be empty)
 - `EV_RC - TYPE INT4`: Specifies whether the method call was successful.
- Instance method `GET_OBJECT_CLASSIFICATIONS` (from SPAM update 63)
 This method can be used to define behavior during the add-on uninstallation for objects and table contents or to define a behavior that differs from that specified by the framework. This means that an add-on can be uninstalled even though not all of the add-on's objects and table contents are supported by the uninstallation framework. Only the data of the add-on to be deleted is taken into account. External objects or table entries will not be taken into account.
 The completeness of the method can be verified during the object list check. Errors in check Add-on uninstallation: Non-deletable objects (`CL_DLVOLC_ADDON_DEL_CHECK`) can be fixed by using the plug-in class and implementation in method `GET_OBJECT_CLASSIFICATIONS`. Implementing the plug-in class and method is not sufficient, however. In addition, the check must be informed about which plug-in class is to be used in the check. To do this, set the parameter value in the OLC checks for the specific delivery, the check, and the attribute `DEINSTALL_PLUGIN` to the name of the plug-in class. For details about parameters in object list checks, see [Specifying Parameters for Object List Checks \[page 85\]](#). The interface has the following properties:
 - Export parameter:
 - `ET_OBJDEF - TYPE TT_OCSOBJDEF`: Table with the uninstallation behavior of the objects and table entries
 - `ET_MESSAGES - TYPE SPROT_U_TAB`: Table of messages (can be empty)
 - `EV_RC - TYPE INT4`: Specifies whether the method call was successful.

Controlling the Uninstallation Behavior

- For objects
 To define the uninstallation behavior of an object, you can define the object as to be deleted or to be ignored.

❖ Example

Examples for objects (parameter `ET_OBJDEF`)

- Object `R3TR PROG RSAD3` is not deleted during uninstallation; it remains in the system.

Field	Value
<code>PGMID</code>	<code>'R3TR'</code>
<code>OBJECT</code>	<code>'PROG'</code>
<code>OBJECT_NAME</code>	<code>'RSAD3'</code>
<code>OPERATION</code>	<code>'I'</code>

- Object `R3TR PROG RSAD5` is deleted during uninstallation.

Field	Value
<i>PGMID</i>	'R3TR '
<i>OBJECT</i>	' PROG '
<i>OBJECT_NAME</i>	' RSAD5 '
<i>OPERATION</i>	' D '

- All objects of object type '*PROG*', which are entered using template */FS/REP/** are deleted.

Field	Value
<i>PGMID</i>	'R3TR '
<i>OBJECT</i>	' PROG '
<i>OBJECT_NAME</i>	' /FS/REP* '
<i>OPERATION</i>	' D '

i Note

The wildcard character '*' is only permitted in field *OBJECT_NAME*. It always has to be right-aligned.

! Restriction

Some object types in the SAP system are delivered with standard uninstallation behavior (see [Handling Object Types \[page 118\]](#)):

- 'D' (Deleted): Object is deleted. This behavior can be overridden in using the plug-in class.
- 'I' (Retained): Object cannot be deleted. This behavior cannot be overridden.
- 'O' (Optional): The object remains in the system. This behavior can be overridden in using the plug-in class.

This uninstallation behavior can only be configured to be weaker, but not stronger than the standard. For objects without standard SAP-defined uninstallation behavior, the method must be used to define this behavior. The uninstallation behavior of object types can also be changed retroactively. If the behavior defined in the method violates the standard behavior, the system will ignore the setting. See the corresponding information in the action log of the SAP Add-On Installation Tools.

- For table entries

To define the uninstallation behavior of a table entry, you can define the table entry as to be deleted or to be ignored. Here, only those table entries are deleted or ignored that were part of the original delivery scope.

i Note

The standard uninstallation behavior defined by SAP for table contents is treated as a suggestion only. The behavior for table entries transferred in the method overwrites the standard SAP-defined behavior.

❖ Example

Examples for table entries (parameter *ET_OBJDEF*)

- The data records with table key **AF*** selected from table **AGR_TIMEB** remain in the system. Only the data records delivered with the add-on to be uninstalled are taken into account.

Field	Value
<i>PGMID</i>	'R3TR'
<i>OBJECT</i>	'TABU'
<i>OBJECT_NAME</i>	'AGR_TIMEB'
<i>TABKEY</i>	'AF*'
<i>OPERATION</i>	'I'

- The data records with table key **DE*** selected from table **'EMCHECK'** are deleted in the system. Only the data records delivered with the add-on to be uninstalled are taken into account.

Field	Value
<i>PGMID</i>	'R3TR'
<i>OBJECT</i>	'TABU'
<i>OBJECT_NAME</i>	'EMCHECK'
<i>TABKEY</i>	'DE*'
<i>OPERATION</i>	'D'

- Objects of object types *VDAT*, *TDAT* and *CDAT* are deleted via their table entries. The associated object must be specified. If you want to handle table entries for these object types in different ways, create a message on the component **BC-UPG-OCS**.

Field	Value
<i>PGMID</i>	'R3TR'
<i>OBJECT</i>	'CDAT'
<i>OBJECT_NAME</i>	'/ABC/V_XYZ'
<i>OPERATION</i>	'D'

❖ Example

Implementation

```
FIELD-SYMBOLS <ls_objdef> LIKE LINE OF et_objdef.
* Object:
  INSERT INITIAL LINE INTO TABLE et_objdef ASSIGNING <ls_objdef>.
  <ls_objdef>-pgmid      = 'R3TR'.
  <ls_objdef>-object     = 'PROG'.
  <ls_objdef>-object_name = 'RSAD5'.
  <ls_objdef>-tabkey     = space.
  <ls_objdef>-operation  = 'D'.
* Key
  INSERT INITIAL LINE INTO TABLE et_objdef ASSIGNING <ls_objdef>.
  <ls_objdef>-pgmid      = 'R3TR'.
  <ls_objdef>-object     = 'TABU'.
```

```
<ls_objdef>-object_name = 'EMCHECK'.  
<ls_objdef>-tabkey      = 'DE*'.  
<ls_objdef>-operation   = 'D'.
```

12.9 Handling Object Types

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 full objects.

❖ Example

The subobject LIMU FUNC xyz is replaced by R3TR FUGR abc and the language part LANG DOMA xyz is replaced by R3TR DOMA xyz.

The tests are executed for the full objects.

A consolidated list of objects that can be deleted in SPAM/SAINT Version 0058 (and higher) can be found under <https://help.sap.com/aak> ► *Application Help* ► *Deletable Objects* ► and in an attachment of SAP Note 2011192 📎

i Note

This list is available in English only. It is updated with every SPAM update.

13 Additional Information

The following is a list of additional helpful topics for users of SAP Add-On Assembly Kit that were already referenced in this documentation.

- [Compatibility of SAP_BASIS Releases \[page 119\]](#)
- [Overview: Import Tools \[page 121\]](#)
- [Modifications and Their Consequences \[page 121\]](#)
- [Conflicts \[page 128\]](#)
- [Examples: Attributes in Software Delivery Assembler \[page 134\]](#)
- [CDs for Add-On Deliveries \[page 136\]](#)
- [Troubleshooting \[page 138\]](#)
- [Further Documentation \[page 139\]](#)
- [Terminology \[page 148\]](#)

13.1 Compatibility of SAP_BASIS 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 release of software component SAP_BASIS can also run on higher downward-compatible SAP_BASIS releases. This means that a package (support package, installation package and upgrade package) attributed to a specific SAP_BASIS release can be installed in higher downward-compatible SAP_BASIS 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_BASIS release cannot be installed in the other SAP_BASIS release. It is not possible to give packages attributes and deliver them in such a way that they can be imported into both SAP_BASIS releases.
-  compatible with restrictions:
In standard cases, packages in this SAP_BASIS release cannot be installed in the other SAP_BASIS release. If it is possible to establish that the ABAP software can run in both SAP_BASIS releases, it is possible to give the packages attributes enabling them to be imported into both SAP_BASIS releases. The import behavior for all SAP_BASIS target 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

SAP_BASIS target release. Here, the original SAP_BASIS release must be removed from the alternative import condition and the higher downward-compatible SAP_BASIS release entered instead.

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

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

Compatibility of SAP_BASIS Releases

SAP_BASIS	620	640	700	701	702	710	711	720	730	731	740	750	751	752	753	754
620																
640																
700																
701																
702																
710																
711																
720																
730																
731																
740																
750																
751																
752																
753																
754																

❖ Example

- A package was built for SAP_BASIS 701. This package can be imported into the SAP_BASIS releases 701, 702, 731, 740 and 750.
- A package was built for SAP_BASIS 710. This package can be imported into the SAP_BASIS releases 710 and 711. If the appropriate attributes are set, it is also possible to import the package into the SAP_BASIS releases 720, 730, 731, 740 and 750.

- Your system has the release level SAP_BASIS 731. Packages for SAP_BASIS releases 700, 701, 702 and 731 can be imported in standard cases. In the case of packages built in the SAP_BASIS releases 710, 711, 720 and 730, the alternative import conditions must be checked for importability.

13.2 Overview: Import Tools

The following is a brief overview of the import tools you can use to import add-on packages.

Tool	Description
SAP Add-On Installation Tool (transaction SAINT)	Installs and upgrades add-on packages directly from the SAP system. Detailed documentation about this can be found in the tool itself.
Support Package Manager (transaction SPAM)	Imports support packages (such as add-on support packages or conflict resolution transports) Detailed documentation about this can be found in the tool itself.
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. For the different scenarios of the SAP system upgrade using SUM, you can find links to the appropriate guides for the relevant SL toolset under https://support.sap.com/sltoolset . You can find individual upgrade guides by using the Guide Finder for SAP NetWeaver and ABAP Platform .

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 \[page 44\]](#).

- **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 \[page 129\]](#).
- **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.
- 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. For further information, see [Additional Information About Enhancements and Modifications \[page 59\]](#).
- If modifications are essential, note the [rules for developing a modifying add-on \[page 125\]](#). Note also the information under [Conflicts \[page 128\]](#).

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:

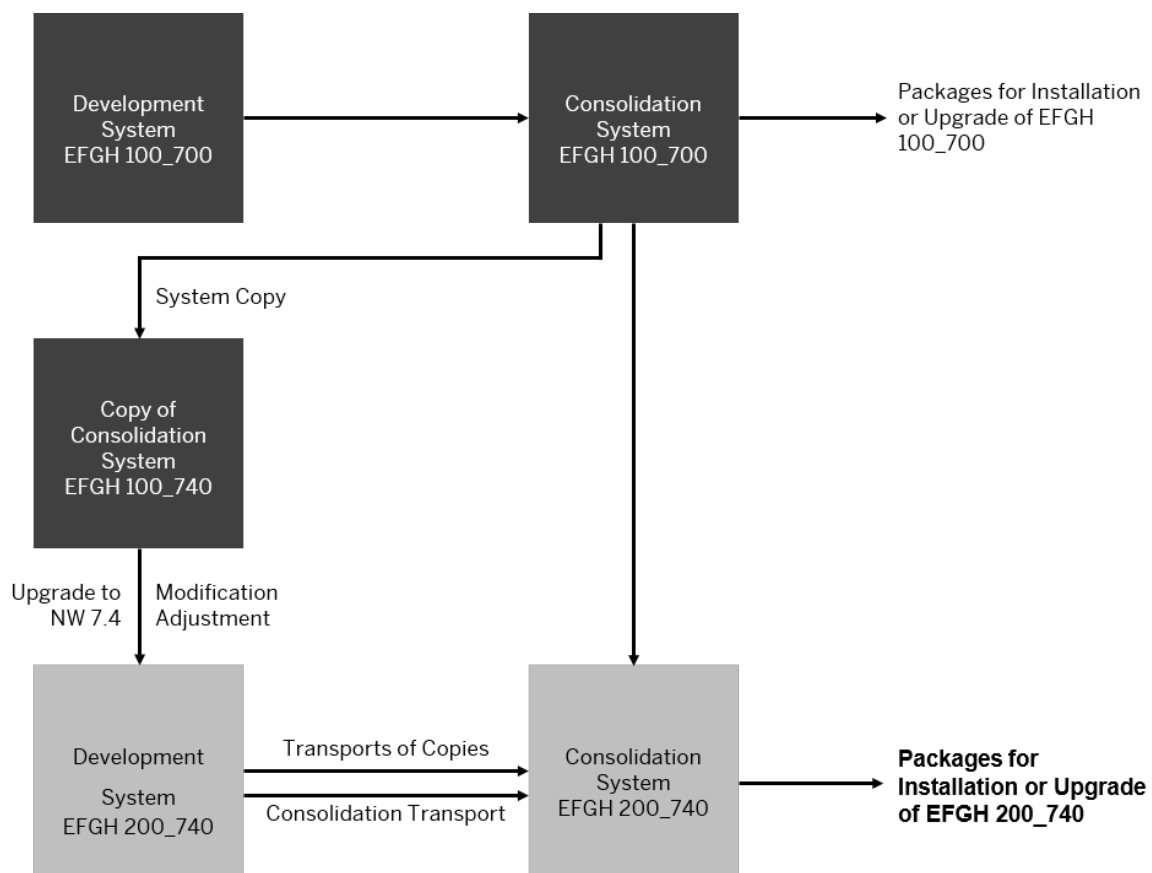
Procedure

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 \[page 48\]](#) 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 \[page 47\]](#) in the new consolidation system and perform the required [settings \[page 48\]](#).
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 about this, see Setting Up and Coordinating Translation (BC-DOC-TTL) under [Further Documentation \[page 139\]](#) ► [Documentation and Translation in SAP Systems](#) ►.
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 \[page 53\]](#)).
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.



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 \[page 29\]](#), 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 \[page 46\]](#).

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 \[page 41\]](#).

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. You need to keep several rules in mind.

- 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 [Further Documentation \[page 139\]](#) ► [The Modification Assistant](#) ►.
- Also note the further restrictions in this table:

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
Modifications to standard SAP customizing	Not allowed	Not allowed
Modifications to domains	Not allowed	Not allowed
Customer exits	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)
Use of customizing includes and system includes	Not allowed	Not allowed

→ Recommendation

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

13.3.4 Creating Conflict Resolution Transports

Conflict Resolution Transports (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.

Context

i Note

CRTs are not required for non-modifying development projects.

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 \[page 25\]](#)).

i Note

Note also the restrictions that apply to permitted changes. You can find these in [Rules for Add-On Maintenance \[page 97\]](#).

Procedure

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.

- a. Adjust the direct conflicts.
 - b. Adjust any dependent objects.
 - c. Adjust any copied objects.
 - d. 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:
 - All modification adjustment requests
 - Any transport requests containing corrections to add-on software

Actions for Corrections in the Consolidation System

6. 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).
7. Use Support Package Manager to import these support packages into the consolidation system and execute transaction `SPDD` (adjust dictionary objects) again if required.
8. 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.
9. 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 any conflicts for repository objects and logical transport objects. It **does not** check for conflicts between table entries.
10. If you need to create a CRT, create the delivery using the SAP Add-On Assembly Kit tools (see [Creating a Delivery \[page 67\]](#)). Create the change list of the CRT by including the transport requests imported into the consolidation system:
 - Transport requests with modification adjustment
 - 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 (namely changes to a software component other than your own), conflicts can occur when the add-on is imported.

Conflicts may arise in the following cases:

- At installation time: When you install and upgrade an add-on using the SAP Add-On Installation Tool
- At maintenance time: When you import Support Packages using Support Package Manager

When packages are imported, the SAP Add-On Installation tool and Support Package Manager check whether there are any conflicts with previously imported add-ons or Support Packages. Conflicts arise only when modifying add-ons are imported. If, for example, an add-on modifies an object that is then delivered in an SAP Support Package, the provider of this add-on must resolve this conflict. If not, the Support Package cannot be imported in the system. If you are a provider of an add-on that contains modifications, you must handle all conflicts to ensure that your add-on works properly. Conflict resolution as described here is not possible in every case and it takes time and effort when it is feasible.

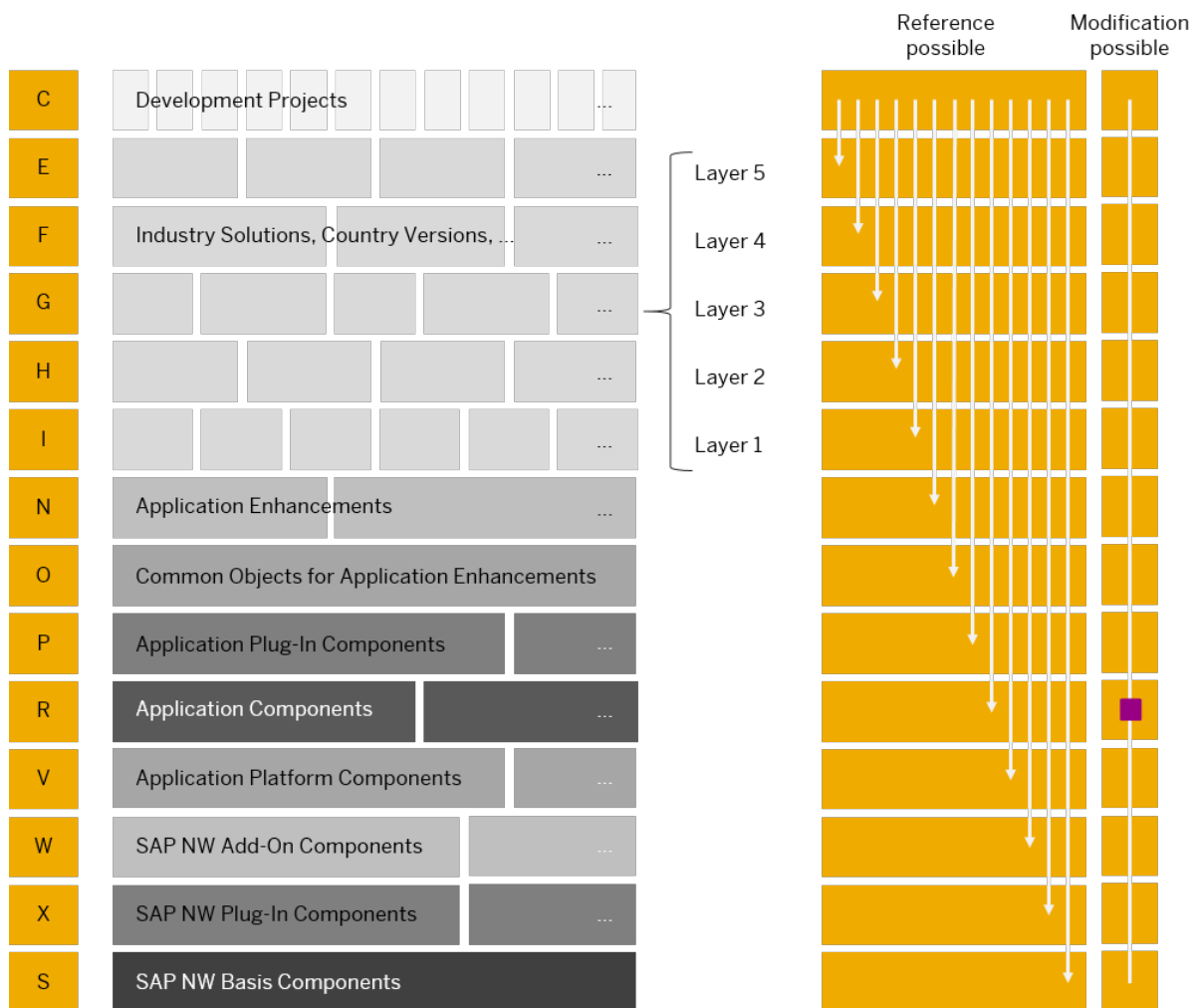
This document contains information about the following topics:

- [Background Information: Conflict Check \[page 128\]](#)
- [Conflicts When Installing/Upgrading an Add-On \[page 129\]](#)
- [Enhancements to Imports of Support Packages \[page 133\]](#)

13.4.1 Background Information: Conflict Check

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 the application component layer 'R'.

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'.



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, the SAP Add-On Installation Tool checks for the following conflicts:

Conflicts with Add-On Components in Lower Layers

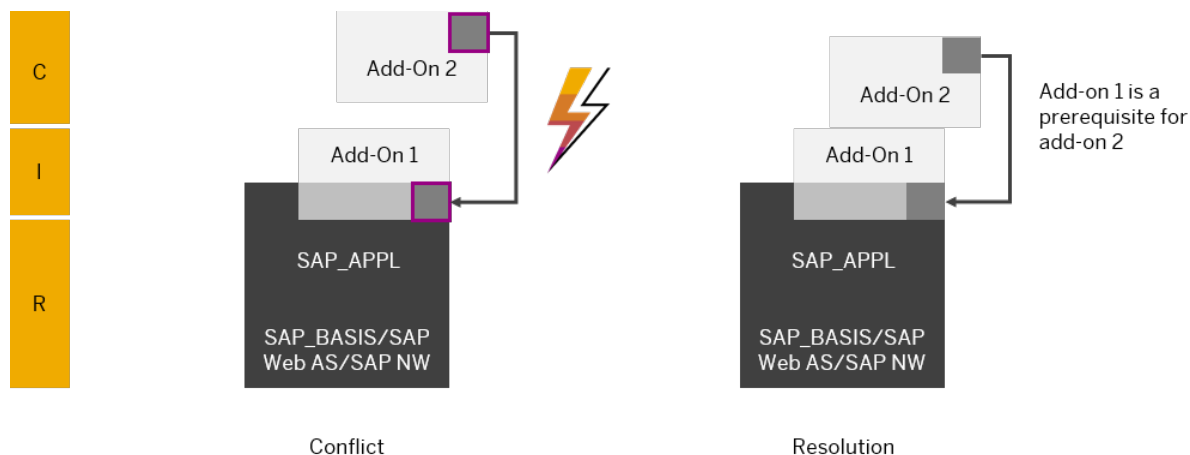
The SAP Add-On Installation Tool checks for conflicts with add-on components in lower layers.

❖ 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, the SAP 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 in Lower Layers

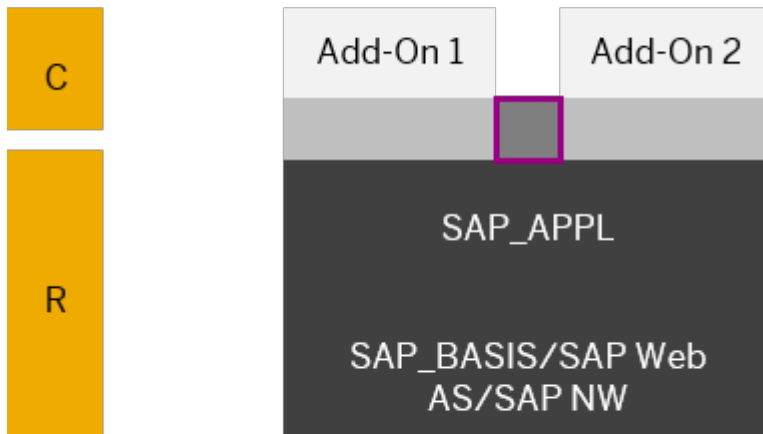
Conflicts with Add-On Components from the Same Layer

The SAP 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, the SAP 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 cannot be resolved

Conflicts with Add-On Components from the Same Layer

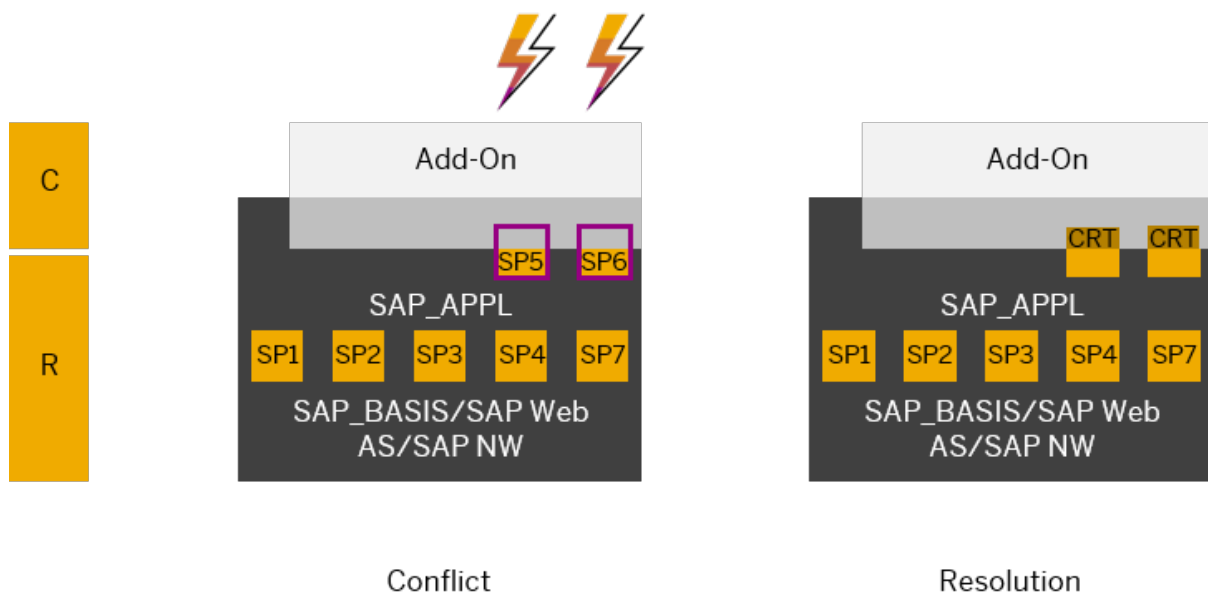
Conflicts with Support Packages in Lower Layers

The SAP 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. The SAP 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, the SAP Add-On Installation Tool detects conflicts with Support Packages 5 and 6. It then prompts the user to add a Conflict Resolution Transport (CRT) 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 \[page 126\]](#).



Conflicts with Support Packages in Lower Layers

Conflicts with Add-On Components in Higher Layers

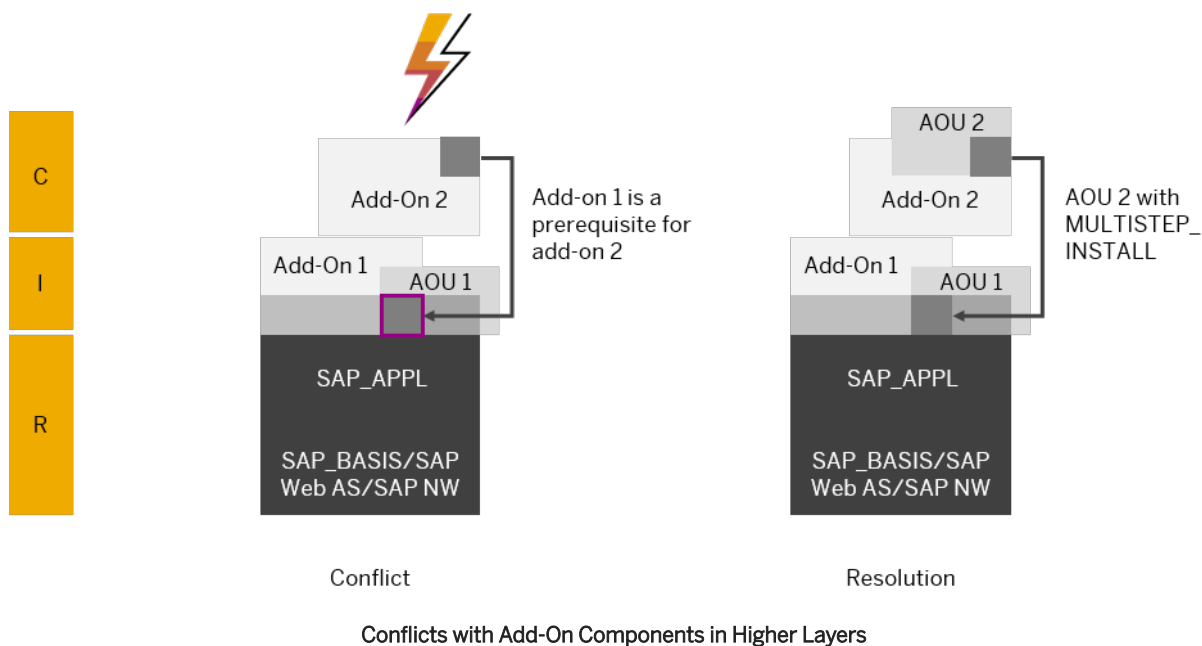
During add-on installation, the SAP Add-On Installation Tool checks for conflicts with add-on components in higher layers.

❖ 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, the SAP 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 the SAP 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.



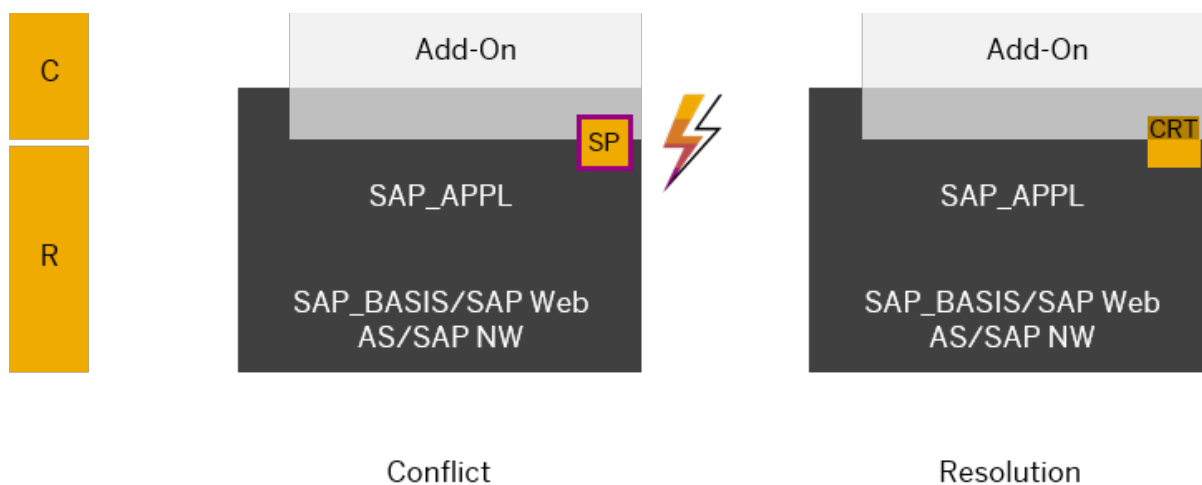
13.4.3 Conflicts when Importing Support Packages

When Support Packages are imported, Support Package Manager checks whether there are any conflicts with add-on components in higher layers.

❖ Example

An add-on in layer 'C' modifies an object in component SAP_APPL (in layer 'R').

If an SAP_APPL Support Package contains corrections to modified objects, Support Package Manager detects a conflict and prompts the user to include a Conflict Resolution Transport in 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 \[page 126\]](#).



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 \[page 134\]](#)
- [Examples: Attributes for Enhancement Packages \[page 135\]](#)
- [Examples: Attributes for SAP HANA \[page 135\]](#)

13.5.1 Examples: Attributes for Add-On Uninstallations

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

Examples of Attributes Set for Uninstallable Add-Ons

Attribute	Value	Comment
<code>DEINSTALL_ALLOWED</code>	1883223	This is the number of the global SAP Note. Add any add-on-specific information to your add-on documentation.
<code>DEINSTALL_PLUGIN</code>	<code>/FINA/CL_FINA_DEINST</code>	<p>The ABAP class must comply with the implementation instructions (see also Plug-In Interface for Add-Ons [page 113]).</p> <p>This attribute is optional. You can use it implement a check class for your add-on.</p>
<code>NEED_SPAM_LEVEL</code>	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> <div><p>→ Recommendation</p><p>We recommend using this attribute to ensure that your customers always have the correct version of the SPAM/SAINT update.</p></div>

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

For more information about these attributes, see [Attributes for Uninstallations \[page 112\]](#).

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.

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-ons 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 *Transport Scenarios for SAP HANA Content* under [Further Documentation \[page 139\]](#) .

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

In this section, you learn how an installation or add-on upgrade CD is structured and how to create packed CD delivery data and unpack SAR archives.

Structure of an Installation CD/Add-On Upgrade CD

Structure of a CD with an Add-On for SAP Release

Define the structure of the CD as follows:

- **DATA**
Save the SAR archive for the add-on package in question in this directory.
- **LANGUAGE**
You can save language-dependent data in this directory, if required.
- **PATCHES**
You can save the SAR archives for the add-on support packages in question in this directory, if required
- **DOCU**
In this directory, the technical documentation about the import process and add-on-specific documentation storage (see also: [Template: Installation Guide \[page 89\]](#)).
- **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.

i Note

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

Structure of a CD with an 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 an 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)

❖ Example

README.ASC

```
*-----* This CD contains the following files:
-----*
README.ASC This file in ASCII-format
LABEL.ASC Media identifier in ASCII-format
DATA/700/AAK_INST_500_700.SAR Archive of Add-On Installation Data for
AOFTOOLS 500_700
DATA/710/AAK_INST_500_710.SAR Archive of Add-On Installation Data for
AOFTOOLS 500_710
DATA/700/AAK_EX_UPG_500_700.SAR Archive of Add-On Exchange Upgrade Data for
AOFTOOLS 500_700
DATA/710/AAK_EX_UPG_500_710.SAR Archive of Add-On Exchange Upgrade Data for
AOFTOOLS 500_710
DOCU/AAK_500_DE.PDF Documentation Add-On Assembly Kit 5.00 in German
DOCU/AAK_500_EN.PDF Documentation Add-On Assembly Kit 5.00 in English
Important
SAP Notes:
-----
350361 Release strategy for Add-On Assembly Kit
SAP AG Walldorf / Germany
```

Creating 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.

i Note

*SAR archives can be created using the `SAPCAR` tool. The `SAPCAR` tool is located in every hardware directory (such as HP-UX, AIX) 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 directory EPS/out to 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.

Unpacking SAR archives

The command for manually unpacking SAR archives for Unix 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 SAP Add-On Installation Tool . This also uploads the SAR archive and unpacks it at the same time. Choose one of the following menu options:

Import Tool	Menu Option
Support Package Manager	Support Package > Load Packages > From Front End
SAP 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 Further Documentation

The following guides you to more detailed information in the different SAP Netweaver releases in SAP Help Portal and in SAP Support Portal.

Documentation in SAP Support Portal

Topic Area	Path to Information in SAP Support Portal
Installation of SAP systems	https://support.sap.com/sltoolset ► System Provisioning ► Installation Options
System copy	https://support.sap.com/sltoolset ► System Provisioning ► System Copy Option
Namespaces	https://support.sap.com/namespaces
SAP Software Download Center	https://support.sap.com/swdc

Documentation in SAP Help Portal

Topic Area	Release	Path to Information in SAP Help Portal
Change and Transport System (Transport Management System, language transport)	SAP NW 7.5	Change and Transport System
	SAP NW 7.4	Change and Transport System
	SAP NW 7.3 EHP1	► Application Help ► SAP NetWeaver Library: Function-Oriented View ► Solution Life Cycle Management ► Software Logistics ► Change and Transport System
	SAP NW 7.3	
	SAP NW 7.0 EHP 3	► Application Help ► SAP NetWeaver by Key Capability ► Solution Life Cycle Management by Key Capability
	SAP NW 7.0 EHP 2	
	SAP NW 7.0 EHP1	► Software Life Cycle Management
	SAP NW 7.0	► Software Logistics ► Change and Transport System
	SAP NW 7.5	► Application Help ► SAP NetWeaver Library: Function-Oriented View
	SAP NW 7.4	► Solution Life Cycle Management
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 7.3 EHP1	► Software Logistics
	SAP NW 7.3	
	SAP NW 7.0 EHP 3	► Application Help ► SAP NetWeaver by Key Capability ► Solution Life Cycle Management by Key Capability
	SAP NW 7.0 EHP 2	
	SAP NW 7.0 EHP1	► Software Life Cycle Management

Topic Area	Release	Path to Information in SAP Help Portal
Documentation and translation in SAP systems	SAP NW 7.0	➤ <i>Software Logistics</i> ➤ or <i>Software Maintenance</i>
	SAP NW 7.5	▮ Application Help ➤ SAP NetWeaver Library: Function-Oriented View
	SAP NW 7.4	➤ Application Server ➤ Application Server
	SAP NW 7.3 EHP1	ABAP ➤ Other Services ➤ Services for Information Developers and Translators ➤
	SAP NW 7.3	
	SAP NW 7.0 EHP 3	▮ Application Help ➤ SAP NetWeaver by Key Capability ➤ Application Platform by Key Capability ➤ Application Server ABAP ➤ Other Services ➤ Services for Information Developers and Translators ➤
	SAP NW 7.0 EHP 2	▮ Application Help ➤ SAP NetWeaver by Key Capability ➤ Application Platform by Key Capability ➤ ABAP Technology ➤ Documentation and Translation Tools (BC-DOC) ➤
	SAP NW 7.0 EHP1	
	SAP NW 7.0	
	SAP NW 7.5	▮ Application Help ➤ SAP NetWeaver Library: Function-Oriented View
	SAP NW 7.4	➤ Application Server ➤ Application Server
	SAP NW 7.3 EHP1	ABAP ➤ Application Development on AS ABAP ➤
	SAP NW 7.3	
ABAP programming basics	SAP NW 7.0 EHP 3	▮ Application Help ➤ SAP NetWeaver by Key Capability ➤ Application Platform by Key Capability ➤ Application Server ABAP ➤ Application Development on AS ABAP ➤
	SAP NW 7.0 EHP 2	▮ Application Help ➤ SAP NetWeaver by Key Capability ➤ Application Platform by Key Capability ➤ ABAP Technology ➤
	SAP NW 7.0 EHP1	
	SAP NW 7.0	
	SAP NW 7.5	▮ Application Help ➤ SAP NetWeaver Library: Function-Oriented View
	SAP NW 7.4	➤ Application Server ➤ Application Server
	SAP NW 7.3 EHP1	ABAP ➤ Application Development on AS ABAP ➤
	SAP NW 7.3	
Package concept	SAP NW 7.5	▮ Application Help ➤ SAP NetWeaver Library: Function-Oriented View
	SAP NW 7.4	➤ Application Server ➤ Application Server
	SAP NW 7.3 EHP1	ABAP ➤ Application Development on AS ABAP ➤ ABAP Workbench - Classic ➤ ABAP Workbench Tools

Topic Area	Release	Path to Information in SAP Help Portal
	SAP NW 7.3	➤ <i>The Package Builder and the ABAP Package Concept</i> ➤
	SAP NW 7.0 EHP 3	▮ Application Help ➤ SAP NetWeaver by Key Capability ➤ Application Platform by Key Capability ➤ Application Server ABAP ➤ Application Development on AS ABAP ➤ ABAP Workbench - Classic ➤ ABAP Workbench Tools ➤ The Package Builder and the ABAP Package Concept ➤
	SAP NW 7.0 EHP 2	▮ Application Help ➤ SAP NetWeaver
	SAP NW 7.0 EHP1	by Key Capability ➤ Application
	SAP NW 7.0	Platform by Key Capability ➤ ABAP Technology ➤ ABAP Workbench (BC-DWB) ➤ ABAP Workbench Tools ➤ Package Builder ➤
	SAP NW 7.5	▮ Application Help ➤ SAP NetWeaver
	SAP NW 7.4	Library: Function-Oriented View
	SAP NW 7.3 EHP1	➤ Application Server ➤ Application Server
	SAP NW 7.3	ABAP ➤ Other Services ➤ Services for Administrators ➤ Customizing (BC-CUS) ➤
	SAP NW 7.0 EHP 3	▮ Hilfe zur Anwendung ➤ SAP
BC Sets and IMG enhancements	SAP NW 7.0 EHP 2	NetWeaver by Key Capability ➤ Solution
	SAP NW 7.0 EHP1	Life Cycle Management by Key
	SAP NW 7.0	Capability ➤ Customizing ➤
	SAP NW 7.5	▮ Application Help ➤ SAP NetWeaver
	SAP NW 7.4	Library: Function-Oriented View
	SAP NW 7.3 EHP1	➤ Security ➤ Identity Management
	SAP NW 7.3	➤ User and Role Administration of Application Server ABAP ➤
	SAP NW 7.0 EHP 3	▮ Application Help ➤ SAP NetWeaver
Roles and authorizations (See also SAP Note 395083 )	SAP NW 7.0 EHP 2	by Key Capability ➤ Security ➤ Identity
	SAP NW 7.0 EHP1	Management ➤ User and Role
	SAP NW 7.0	Administration of Application Server
	SAP NW 7.5	ABAP ➤
	SAP NW 7.4	
	SAP NW 7.3 EHP1	
	SAP NW 7.3	
	SAP NW 7.0 EHP 3	

Topic Area	Release	Path to Information in SAP Help Portal
Delivery class of tables	SAP NW 7.5	► Application Help ► SAP NetWeaver Library: Function-Oriented View
	SAP NW 7.4	► Application Server ► Application Server ABAP ► Application Development on AS ABAP ► ABAP Workbench - Classic ► ABAP Dictionary ► Database Tables ► Specifying Database Tables ► Creating Database Tables and Table Fields ►
	SAP NW 7.3 EHP1	► Application Help ► SAP NetWeaver Library: Function-Oriented View
	SAP NW 7.3	► Application Server ► Application Server ABAP ► Application Development on AS ABAP ► ABAP Workbench - Classic ► ABAP Dictionary ► Tables ► Creating Tables ► Delivery Class ►
	SAP NW 7.0 EHP 3	► Application Help ► SAP NetWeaver by Key Capability ► Application Server ► Application Server ABAP ► Application Development on AS ABAP ► ABAP Workbench - Classic ► ABAP Dictionary ► Tables ► Creating Tables ► Delivery Class ►
	SAP NW 7.0 EHP 2	► Application Help ► SAP NetWeaver by Key Capability ► Application Platform by Key Capability ► ABAP Technology ► ABAP Workbench (BC-DWB) ► BC - ABAP Dictionary ► Tables ► Creating Tables ► Delivery Class ►
	SAP NW 7.0 EHP1	
	SAP NW 7.0	
Changing the SAP Standard (BC)	SAP NW 7.5	► Application Help ► SAP NetWeaver Library: Function Oriented View
		► Application Server ► Application Server ABAP ► Application Development on AS ABAP ► Customer-Specific ABAP Development ► Changing the SAP Standard (BC) ►
	SAP NW 7.4	► Application Help ► SAP NetWeaver Library: Function Oriented View
	SAP NW 7.3 EHP1	► Application Server ► Application Server

Topic Area	Release	Path to Information in SAP Help Portal
	SAP NW 7.3	ABAP > Application Development on AS ABAP > ABAP Customer Development > Changing the SAP Standard (BC) >
	SAP NW 7.0 EHP 3	Application Help > SAP NetWeaver by Key Capability > Application Server > Application Server ABAP > Application Development on AS ABAP > ABAP Customer Development > Changing the SAP Standard (BC) >
	SAP NW 7.0 EHP 2	Application Help > SAP NetWeaver by Key Capability > Application Platform by Key Capability > ABAP Technology > ABAP Workbench (BC-DWB) > Changing the SAP Standard (BC) >
	SAP NW 7.0 EHP1	
	SAP NW 7.0	
The Modification Assistant	SAP NW 7.5	Application Help > SAP NetWeaver Library: Function Oriented View > Application Server > Application Server ABAP > Application Development on AS ABAP > Customer-Specific ABAP Development > Changing the SAP Standard (BC) > Modifying the Standard > The Modification Assistant >
	SAP NW 7.4	Application Help > SAP NetWeaver Library: Function Oriented View
	SAP NW 7.3 EHP1	Application Server > Application Server ABAP > Application Development on AS ABAP > ABAP Customer Development > Changing the SAP Standard (BC) > Modifying the Standard > The Modification Assistant >
	SAP NW 7.3	

Topic Area	Release	Path to Information in SAP Help Portal
	SAP NW 7.0 EHP 3	► Application Help ► SAP NetWeaver by Key Capability ► Application Server ► Application Server ABAP ► Application Development on AS ABAP ► ABAP Customer Development ► Changing the SAP Standard (BC) ► Modifying the Standard ► The Modification Assistant ►
	SAP NW 7.0 EHP 2	► Application Help ► SAP NetWeaver by Key Capability ► Application
	SAP NW 7.0 EHP1	Platform by Key Capability ► ABAP Technology ► ABAP Workbench (BC-DWB) ► Changing the SAP Standard (BC) ► The Modification Assistant ►
	SAP NW 7.0	
Custom Code Migration	SAP NW 7.5	► Application Help ► SAP NetWeaver Library: Function Oriented View ► Application Server ► Application Server ABAP ► Application Development on AS ABAP ► Customer-Specific ABAP Development ► Custom Code Migration ►
Enhancement Framework	SAP NW 7.5	► Application Help ► SAP NetWeaver Library: Function Oriented View ► Application Server ► Application Server ABAP ► Application Development on AS ABAP ► Customer-Specific ABAP Development ► Enhancement Framework ►
	SAP NW 7.4	► Application Help ► SAP NetWeaver Library: Function Oriented View
	SAP NW 7.3 EHP1	► Application Server ► Application Server ABAP ► Application Development on AS ABAP ► ABAP Customer Development ► Enhancement Framework ► Enhancement Concept ► Enhancement Spots ►
	SAP NW 7.3	

Topic Area	Release	Path to Information in SAP Help Portal
	SAP NW 7.0 EHP 3	▶ Application Help ▶ SAP NetWeaver by Key Capability ▶ Application Server ▶ Application Server ABAP ▶ Application Development on AS ABAP ▶ ABAP Customer Development Enhancement Framework ▶
	SAP NW 7.0 EHP 2	▶ Application Help ▶ SAP NetWeaver by Key Capability ▶ Application Platform by Key Capability ▶ ABAP Technology ▶ ABAP Workbench (BC-DWB) ▶ Enhancement Framework ▶
	SAP NW 7.0 EHP1	
	SAP NW 7.0	
Switch Framework (As of SAP NetWeaver 7.0, you can use the Switch Framework to enable or disable software components.)	SAP NW 7.5	▶ Application Help ▶ SAP NetWeaver Library: Function Oriented View
	SAP NW 7.4	▶ Application Server ▶ Application Server ABAP ▶ Application Development on AS ABAP ▶ ABAP Workbench - Classic ▶ Switch Framework ▶
	SAP NW 7.3 EHP1	
	SAP NW 7.3	▶ Application Help ▶ SAP NetWeaver Library: Function Oriented View ▶ Application Server ▶ Application Server ABAP ▶ Application Development on AS ABAP ▶ ABAP Programming Tools ▶ Switch Framework ▶
	SAP NW 7.0 EHP 3	▶ Application Help ▶ SAP NetWeaver by Key Capability ▶ Application Server ▶ Application Server ABAP ▶ Application Development on AS ABAP ▶ ABAP Workbench - Classic ▶ Switch Framework ▶
	SAP NW 7.0 EHP 2	▶ Application Help ▶ SAP NetWeaver by Key Capability ▶ Application Platform by Key Capability ▶ ABAP Technology ▶ ABAP Workbench (BC-DWB) ▶ Switch Framework ▶
	SAP NW 7.0 EHP1	
	SAP NW 7.0	

Topic Area	Release	Path to Information in SAP Help Portal
Enhancement spots	SAP NW 7.5	Application Help > SAP NetWeaver Library: Function Oriented View > Application Server > Application Server ABAP > Application Development on AS ABAP > Customer-Specific ABAP Development > Enhancement Framework > Enhancement Concept > Enhancement Spots
	SAP NW 7.4	Application Help > SAP NetWeaver Library: Function Oriented View
	SAP NW 7.3 EHP1	Application Server > Application Server ABAP > Application Development on AS ABAP > ABAP Customer Development > Enhancement Framework > Enhancement Concept > Enhancement Spots
	SAP NW 7.3	Application Server > Application Server ABAP > Application Development on AS ABAP > ABAP Customer Development > Enhancement Framework > Enhancement Concept > Enhancement Spots
	SAP NW 7.0 EHP 3	Application Help > SAP NetWeaver by Key Capability > Application Server > Application Server ABAP > Application Development on AS ABAP > ABAP Customer Development > Enhancement Framework > Enhancement Concept > Enhancement Spots
	SAP NW 7.0 EHP 2	Application Help > SAP NetWeaver by Key Capability > Application Platform by Key Capability > ABAP Technology > ABAP Workbench (BC-DWB) > Enhancement Framework > Enhancement Concept > Enhancement Spots
	SAP NW 7.0 EHP1	
	SAP NW 7.0	
Append structures	SAP NW 7.5	Application Help > SAP NetWeaver Library: Function-Oriented View
	SAP NW 7.4	Application Server > Application Server ABAP > Application Development on AS ABAP > ABAP Workbench - Classic > ABAP Dictionary > Database Tables > Making Changes to Tables > Adding an Append Structure

Topic Area	Release	Path to Information in SAP Help Portal
	SAP NW 7.3 EHP1	▶ Application Help ▶ SAP NetWeaver Library: Function-Oriented View ▶ Application Server ▶ Application Server ABAP ▶ Application Development on AS ABAP ▶ ABAP Workbench - Classic ▶ ABAP Dictionary ▶ Tables ▶ Append Structures ▶
	SAP NW 7.3	▶ Application Help ▶ SAP NetWeaver Library: Function-Oriented View ▶ Application Server ▶ Application Server ABAP ▶ Application Development on AS ABAP ▶ ABAP Programming Tools ▶ ABAP Dictionary ▶ Tables ▶ Append Structures ▶
	SAP NW 7.0 EHP 3	▶ Application Help ▶ SAP NetWeaver by Key Capability ▶ Application Server ▶ Application Server ABAP ▶ Application Development on AS ABAP ▶ ABAP Workbench - Classic ▶ ABAP Dictionary ▶ Tables ▶ Append Structures ▶
	SAP NW 7.0 EHP 2	▶ Application Help ▶ SAP NetWeaver by Key Capability ▶ Application
	SAP NW 7.0 EHP1	Platform by Key Capability ▶ ABAP
	SAP NW 7.0	Technology ▶ ABAP Workbench (BC-DWB) ▶ BC - ABAP Dictionary ▶ Tables ▶ Append Structures ▶

13.9 Terminology

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

Term	Explanation
add-on	<p>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.</p> <p>They are installed using a delivery package, which is imported via the SAP Add-On Installation Tool. The delivery package combines multiple transport request and contains the development objects of the add-on.</p>
add-on delta upgrade	Upgrade the add-on release without upgrading the SAP system
add-on exchange upgrade	Upgrade 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	<p>List of all new and modified objects in Software Delivery Composer relevant for a specific delivery. This includes:</p> <ul style="list-style-type: none">• Change piece list• Supplement change piece list• Support package• CRT
change piece list	In Software Delivery Composer; a 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 <code>SAP_APPL</code> or <code>SAP_HR</code>)
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	<p>In Software Delivery Composer, an information container for creating all transport objects and software objects relevant for a specific delivery. This includes:</p> <ul style="list-style-type: none">• Change list• Component piece list• Supplement component piece list• Exchange component piece list• Exclusive lists

Term	Explanation
delivery component	<p>Set of software objects. A delivery component comprises the following objects:</p> <ul style="list-style-type: none"> • Software object of the modifying software component • Modified objects from a different software component
delivery type	<p>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:</p> <ul style="list-style-type: none"> • Installation/upgrade • Conflict resolution transport • Support package • Pilot
Exchange component piece list	<p>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:</p> <ul style="list-style-type: none"> • From SAP NetWeaver 7.0 to 7.5: add-on exchange package (AOX) • Up to and including SAP Web AS 6.40: Add-on upgrade package (AOU) with import attribute <code>CREATE_FULLTASK=T</code> <p>Exchange component piece lists are no longer used after SAP_BASIS 751.</p>
conflict resolution transport (CRT)	<ol style="list-style-type: none"> 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 <code>SAP_BASIS</code> or <code>SAP_HR</code> . 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.



Term	Explanation
namespace	<p>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.</p> <p>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.</p> <p>A namespace is also required for the add-on software component and when creating deliveries using SAP Add-On Assembly Kit.</p>
namespace role	<p>Defines the origin of the development objects and the type of changes that can be made to objects in this namespace.</p> <p>The role P(for producer) can be chosen only if the development license is known. New objects can be created.</p> <p>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.</p>
package type	<p>Type of an importable package.</p> <p>Software Delivery Assembler converts delivery requests to importable packages. You can use SAP Add-On Assembly Kit to create the following package types:</p> <ul style="list-style-type: none"> • 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	<p>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.</p> <p>Support packages can be created for each software component.</p>
software component version	Version of a software component identified by an add-on ID and add-on release.
support package	<ol style="list-style-type: none"> 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

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.

About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
 - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
 - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.

The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

© 2023 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see <https://www.sap.com/about/legal/trademark.html> for additional trademark information and notices.