Configuration Guide | PUBLIC

2021-09-08

Destruction of Attachments with SAP Information Lifecycle Management

Generic Object Services (GOS), ArchiveLink (ARL) and Business Document Service (BDS)

THE BEST RUN SAP

Content

Document History

Version	Date	Description
1.0	February 09, 2021	Initial version

1 Introduction

To comply with data protection regulations such as the European General Data Protection Regulation (GDPR), all person-related data must be destroyed unless a legitimate purpose for processing or storing that data exists.

The majority of data in an SAP system is structured in database tables and can be managed with SAP Information Lifecycle Management (ILM). This data often has unstructured data attached to it, for example images, PDF documents or any other file type that can be stored in an SAP system. Once the retention period for the leading (structured) data object is over, these attachments must be destroyed together with the object they are linked to.

The central frameworks Generic Object Services (GOS), ArchiveLink (ARL), or Business Document Service (BDS) enable linking unstructured data to business objects (that is, BOR objects). The underlying framework of GOS and BDS is the Knowledge Provider (KPro) framework, which enables the connection to the Content Server infrastructure or to a storage location on the database.

The frameworks all use one or more central tables to store the links between BOR objects and the data file identifiers in the document storage system. Once table data defining the BOR object instance is destroyed, the link to the attachment file must be destroyed as well. In some cases, multiple BOR objects are linked to a single attachment. The attachment file itself must be destroyed as soon as no further links to any BOR object exist.

In section General Information: Destruction of Attachments [page 5] of this document, you get a general overview of how the destruction of attachments (GOS, ARL, BDS) is handled for ILM objects delivered by SAP. For nonstandard ILM objects that you created yourself, refer to section What to Consider with Customer ILM Objects [page 9] to get more information about how these objects can be enabled for attachment destruction.

For an overview of SAP Notes/ Support Packages with which the functionality is available, see section How To Enable the Destruction of Attachments [page 13].

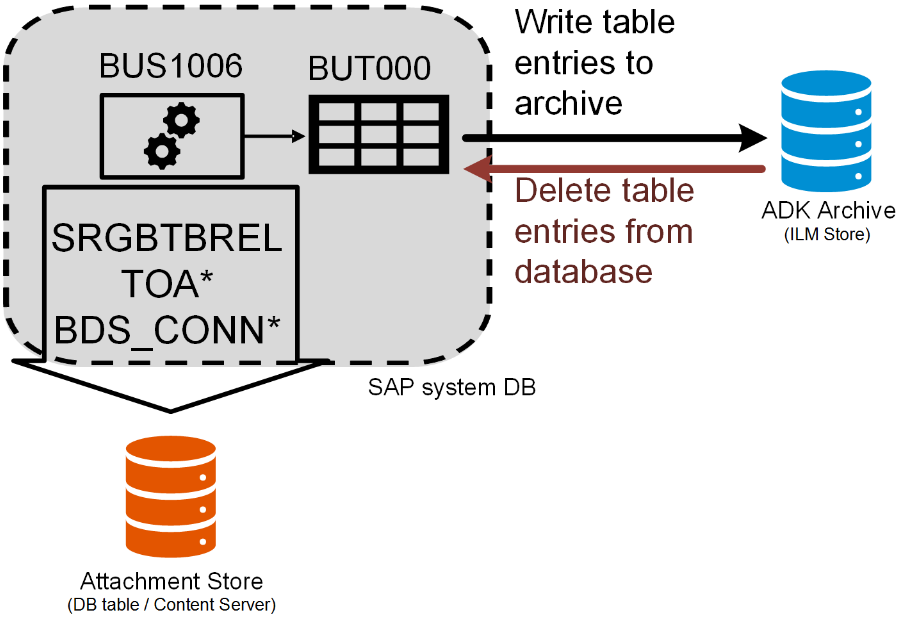
> i Note
>
> The information given in this guide does not apply to Document Management System (DMS) documents that are archived with ILM object `CV_DVS`. These documents inherit the relevant ILM rules from the business object they are linked to.

Related Information

- Data Protection
- SAP Information Lifecycle Management

2 General Information: Destruction of Attachments

Before data is destructed, in most cases it is archived and moved to an ILM Store. Archiving is typically performed on a regular basis to keep the database size at a manageable level. In addition, moving data to the archive restricts data processing. In case of personal data, this might be required to comply with data protection regulations.



BUS1006
BUT000
SRGBTBREL
TOA*
BDS_CONN*
Write table entries to archive
Delete table entries from database
ADK Archive
(ILM Store)
SAP system DB
Attachment Store
(DB table / Content Server)

Figure 1: Archiving of Business Partner Master Data with Attachments

The example in figure 1 shows the archiving process for business partner master data (`BUS1006`) that has attachments assigned. Its main data is stored in table `BUT000` and can be archived using archiving object `CA_BUPA`. Archiving data from the SAP system is a two-step process: First, the data is written to an archive file in the ILM Store (black arrow). Secondly, the data is deleted (red arrow) from the relevant database tables (in this case `BUT000`).

Attachments are stored on the Content Server (orange database icon) and are connected to the relevant BOR objects via links. These links are stored in the respective tables of the attachment frameworks:

- For GOS: Table `SRGBTBREL`
- For ARL: Tables starting with `TOA*`
- For BDS: Tables starting with `BDS_CONN*`

When data is being archived, the link entries remain in the link tables, although the leading data object was written to an archive file and deleted from the database. The same applies to the actual attachment files, which remain in the attachment store. The destruction of both the leading data object and its attachments is possible (and in case of personal data even mandatory) after the retention period is over.

Data destruction is typically performed in transaction ILM_DESTRUCTION. The following ways of destroying data are possible:

- Destruction of Archived Data [page 6]
- Destruction from the Database [page 7]
- Destruction of Attachments and Print Lists. This option refers to the destruction of ArchiveLink attachments referenced to ILM and will not be discussed in this documentation. For more information, see Destroying Data from an ILM Store.

2.1 Destruction of Archived Data

You can schedule the destruction of archive files by choosing *Archive Files (ADK)* from the selection criteria in transaction ILM_DESTRUCTION. For this, you must first define ILM policies and rules and set up a file storage in a certified ILM Store.

After the retention period has expired, archiving sessions can be collected in a destruction worklist. The processing of this worklist is called a destruction run (marked in red). During the destruction run, the BOR object instances contained in each archive file are identified by the ILM framework (see ILM Object-Specific Customizing for Link Identification [page 9]). The system identifies the links to the relevant attachments via the respective implementations of *BAdI: Clean Up Activity Before Deleting an Archive File* (BADI_ILM_PRE_DESTR_FILE_ACTION) and deletes them from the link tables (light green arrow).

As soon as the last link to an attachment was destroyed, the ARL and BDS Frameworks automatically destroy the attachment file in the attachment storage. For GOS attachments, however, the periodical execution of cleanup report *Reorganization of Documents and Send Requests* (transaction RSBCS_REORG) is necessary to delete the orphaned attachment files and metadata.

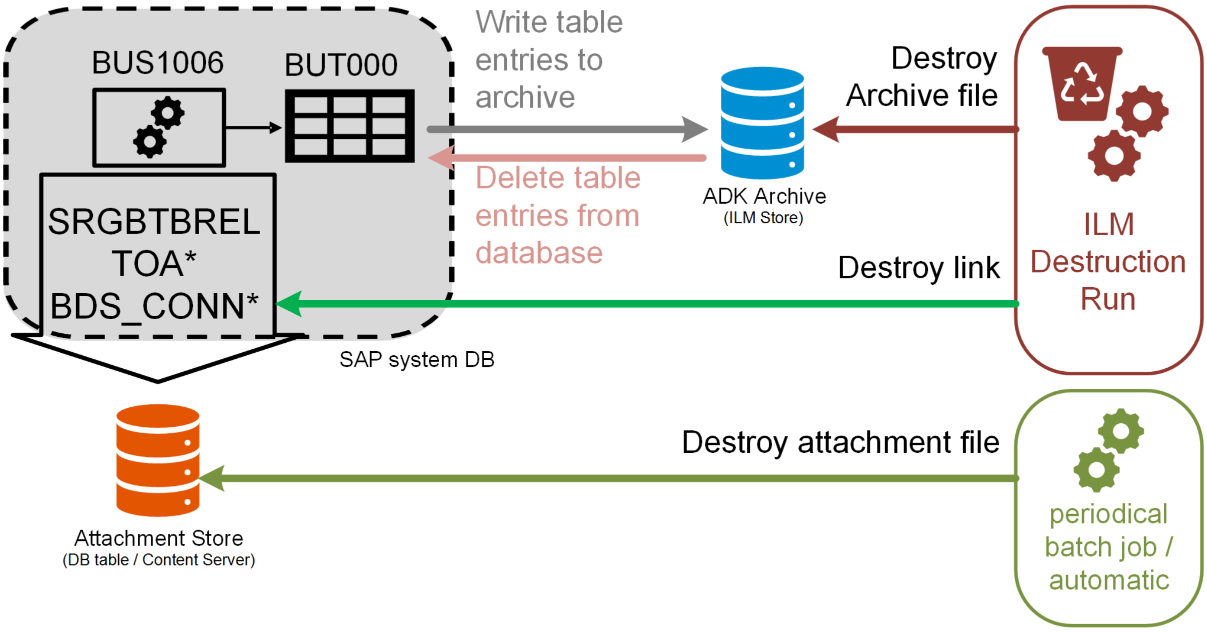


Figure 2: Destruction of Attachments when Destroying Data from Archive Files

Related Information

- ILM Store
- Editing ILM Policies

2.2 Destruction from the Database

From a technical point of view, ILM destruction reuses large parts of the archiving processes. Therefore, to destroy data without archiving it first, select the *Data Destruction* checkbox under *ILM Actions* in the Write program of the relevant archiving object.

As shown in figure 3, a temporary archive file is created, which is deleted after the successful execution of the delete program. The destruction of attachments is performed via implementations of *BadI: Clean Up Activities Prior to Destroying a Data Object with Data Archiving* (`BADI_ILM_PRE_DESTR_OBJ_ACTION`) that exist for each of the attachment frameworks. The implementations are called during the execution of the delete program. The identification of BOR object types and keys (see ILM Object-Specific Customizing for Link Identification [page 9]) is the same as the one for destroying data from the archive.

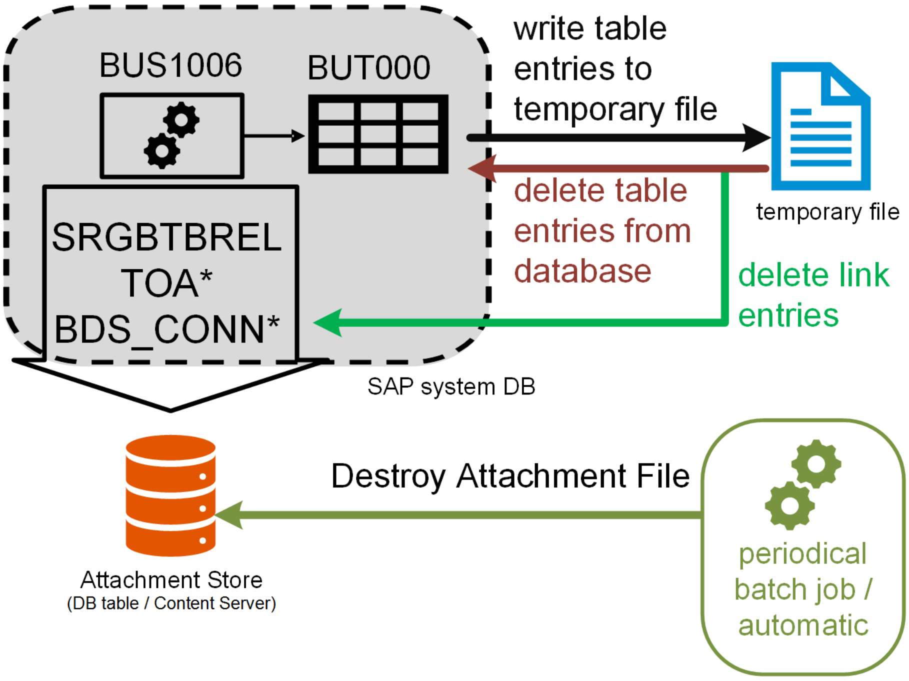

BUS1006
BUT000
write table entries to temporary file
temporary file
delete table entries from database
delete link entries
SRGBTBREL
TOA*
BDS_CONN*
SAP system DB
Destroy Attachment File
periodical batch job / automatic
Attachment Store
(DB table / Content Server)

Figure 3: Destruction of Attachments when Destroying Data from the Database

2.3 Destruction Using ILM Destruction Objects/ Non-ILM Destruction Reports

The destruction of data from the archive or from the database is handled with SAP ILM. Whereas when using destruction objects, the destruction of attachments must be realized by calling the following methods or functions:

- For **Generic Object Services** attachments, call method `DELETE_AL_ITEM` of class `CL_GOS_API`.
- For **ArchiveLink** attachments, call function module `ARCHIV_DELETE_META`.
- For **Business Document Service** attachments, either call method `DELETE` of class `CL_BDS_DOCUMENT_SET` or function module `BDS_BUSINESSDOCUMENT_DELETE`.

3 What to Consider with Customer ILM Objects

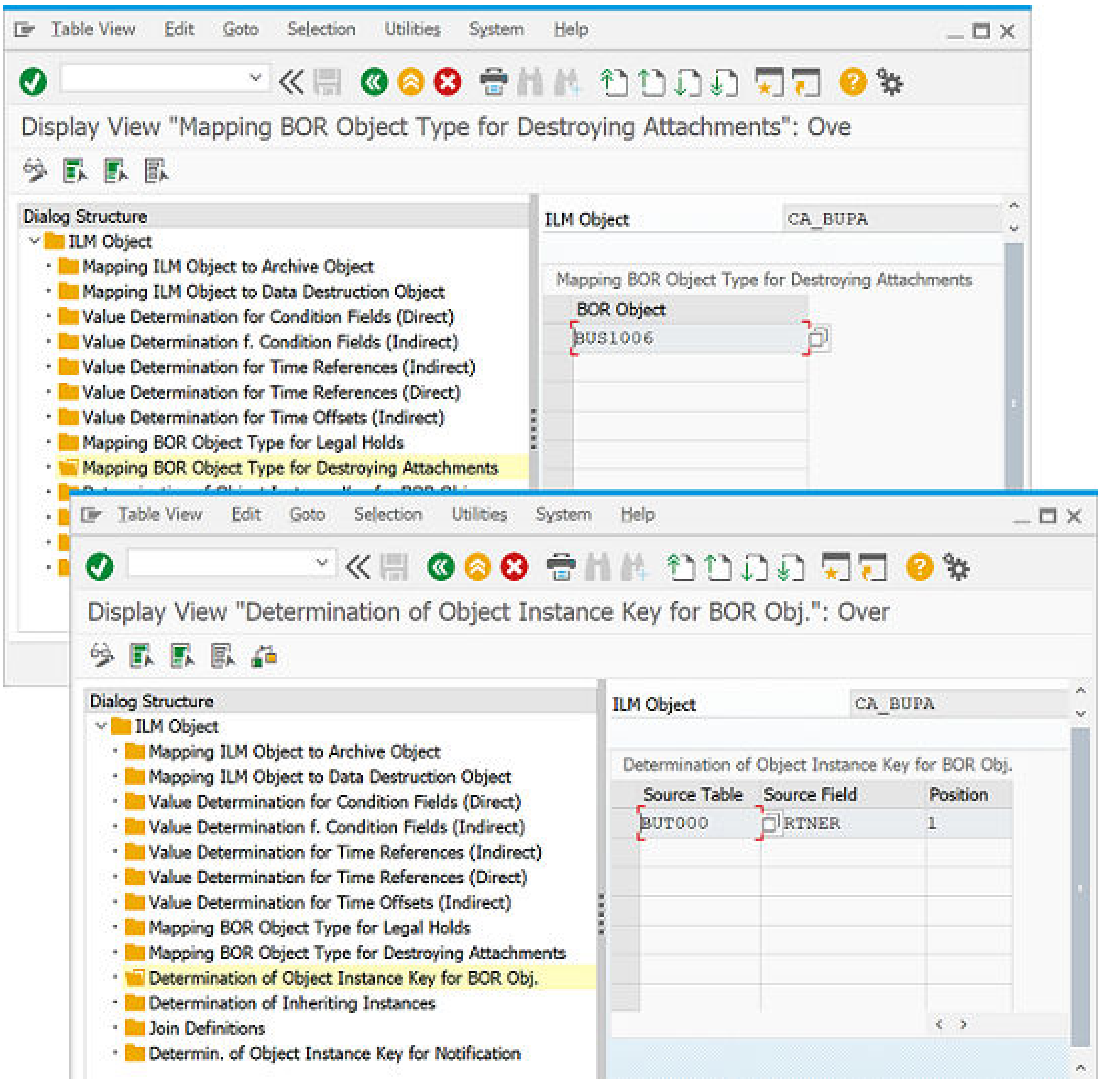
The information given in section General Information: Destruction of Attachments [page 5] applies to SAP-delivered ILM objects. If you have created your own ILM objects, refer to the following sections to find out what additional actions are required to enable these objects for the destruction of attachments.

3.1 ILM Object-Specific Customizing for Link Identification

As discussed in General Information: Destruction of Attachments [page 5]., attachments are connected to their respective BOR objects via links. The regular archiving process is performed for table entries only though. Therefore, you must maintain the following Customizing settings to enable the destruction of attachments.

Let's have a look again at the example given in figure 1: The related BOR object instance (BOR type and key) must be determined. The (archived) table entries are analyzed generically by method `GET_BOR_KEYS_FROM_DATA_OBJECT` of class `CL_ARC_DAS_ROUT_ADK`.

In order for the relevant BOR objects to be identified, you must make the necessary Customizing settings in transaction `IRM_CUST_BS` (see Figure 4). Determine the BOR object type and the BOR object key. The key is created based on the souce table and source field you specify, the field values are concatenated in the specified order to construct the BOR key.



Display View "Mapping BOR Object Type for Destroying Attachments": Ove

Dialog Structure
- ILM Object
  - Mapping ILM Object to Archive Object
  - Mapping ILM Object to Data Destruction Object
  - Value Determination for Condition Fields (Direct)
  - Value Determination f. Condition Fields (Indirect)
  - Value Determination for Time References (Indirect)
  - Value Determination for Time References (Direct)
  - Value Determination for Time Offsets (Indirect)
  - Mapping BOR Object Type for Legal Holds
  - Mapping BOR Object Type for Destroying Attachments

ILM Object CA_BUPA

Mapping BOR Object Type for Destroying Attachments

BOR Object
BUS1006

Display View "Determination of Object Instance Key for BOR Obj.": Over

Dialog Structure
- ILM Object
  - Mapping ILM Object to Archive Object
  - Mapping ILM Object to Data Destruction Object
  - Value Determination for Condition Fields (Direct)
  - Value Determination f. Condition Fields (Indirect)
  - Value Determination for Time References (Indirect)
  - Value Determination for Time References (Direct)
  - Value Determination for Time Offsets (Indirect)
  - Mapping BOR Object Type for Legal Holds
  - Mapping BOR Object Type for Destroying Attachments
  - Determination of Object Instance Key for BOR Obj.
  - Determination of Inheriting Instances
  - Join Definitions
  - Determin. of Object Instance Key for Notification

ILM Object CA_BUPA

Determination of Object Instance Key for BOR Obj.

Source Table	Source Field	Position
BUT000	RTNER	1

Figure 4: Defining the BOR Object Type and Key in ILM Customizing (IRM_CUST_BS)

In most cases, an ILM object is related to a single BOR object type. Multiple BOR object types are only supported, if their key definitions are identical. For more complex ILM objects related to multiple BOR object types and different or varying BOR key definitions, you must implement BadI method `GET_OBJECTS` of *BAdI: Determination of BOR Object Instance Key for Destroying Attachments* (`BADI_IRM_ATTACHMENTS`) must be implemented to construct the BOR key. The standard key determination is skipped as soon as the BAdI implementation provides the key definition.

For examples of such complex cases, refer to the BAdI implemementations created for the customer/ vendor master data ILM objects (`FI_ACCRECV`, `FI_ACCPAYB`) and the ILM object `FI_DOCUMNT`.

3.2 GOS-Specific Requirements

3.2.1 Skipping the BOR Existence Check

The existence check is a nonmandatory method of a BOR object. It determines if a specific BOR instance exists on the database. If the method `EXISTENCECHECK` is not implemented in a BOR object, the existence check always returns the message *Object instance exists (sy-subrc = 0)*.

For GOS attachments, this existence check is used to verify that the leading BOR object was successfully removed from the database. Consequently, the destruction of GOS attachments cannot be performed if the method is not implemented in the respective BOR object. In some cases, it is not possible to implement the existence check due to issues with certain workflows.

By implementing method `SKIP_EXISTENCE_CHECK` of *BAdI: Determination of BOR Object Instance Key for Destroying Attachments* (BADI_IRM_ATTACHMENTS), a flag is added to the list of BOR object instances found by the standard logic or by an implementation of BadI method `GET_OBJECTS`. For all flagged BOR object instances, the existence check used by the GOS framework during the destruction of the attachment links are skipped.

Make sure to implement the following SAP Notes to enable skipping of the BOR existence check:

Table 1: Overview of SAP Notes: Skipping the BOR Existence Check

SAP NetWeaver Component	SAP Note	Description
ILM	2888966	Deletion of Attachments - Skip BOR Instance Existence Check
ADK	2888881	Postprocessing for Data Destruction for Data Archiving (ILM): Existence Check
GOS	2889006	GOS: Skipping of Existence Check During an ILM Destruction Run

3.2.2 Archiving Class SGOS

The archiving of GOS attachments together with their leading data object is possible by including archiving class `SGOS` into the leading archiving object. In addition, the function module `SGOS_ARCHIVE_OBJECT` must be called during the archive write report.

Although this solution is hardly used, it can simplify the handling of attachments during archiving and destruction. However, neither the ARL nor the BDS framework offer a similar archiving class. Therefore, making ILM object-specific customizing setting is still necessary to support the destruction of these attachment types.

3.3 ARL-Specific Requirements

3.3.1 Handling of ArchiveLink References to Attachments and Print Lists

The destruction of ARL attachments and print lists is one of the options in transaction `ILM_DESTRUCTION`. References to the leading BOR objects can be created, which can then inherit the retention rules from the leading object. The destruction of ARL link entries is skipped in case such a reference is found for a particular ARL link.

For more information, refer to the activity diagrams in sections Destruction from the Archive [page 15] and Destruction from the Database [page 16].

3.4 BDS-Specific Requirements

3.4.1 Other Types of Attachments in BDS

Apart from BOR objects, the BDS framework allows you to store other types of attachments. The type of leading entity is defined in the BDS link tables `BDS_CONNxx`: BO (BOR object), CL (Class type), or OT (other type).

Only leading entities of type BO are supported by ILM destruction, as described in General Information: Destruction of Attachments [page 5]. The types CL and OT cannot easily be linked to a clearly defined leading object and therefore must be treated individually by the application. One way to achieve this is by implementing *BadI: Clean Up Activities Before Deleting an Archive File* (`BADI_ILM_PRE_DESTR_FILE_ACTION`) and *BadI: Clean Up Activities Prior to Destroying a Data Object with Data Archiving*(`BADI_ILM_PRE_DESTR_OBJ_ACTION`).

For more information, refer to the system documentation of the BAdIs in transaction `SE18`.

4 How To Enable the Destruction of Attachments

The functionality is available with the following Support Packages and SAP Notes:

Table 2: Overview of SAP Notes: Enablement of Attachment Destruction

Product	Attachment Type	SAP Note	Available as Of
SAP ERP	GOS, ARL, BDS	Central SAP Note 2787893	SAP Enhancement Package 8, Support Package 16 for SAP ERP 6.0
SAP S/4HANA			Refer to the S/4HANA-specific SAP Notes referenced in the central SAP Note
Customer Relationship Management (CRM)	GOS, ARL, KPro	• SAP Note 2658318 • SAP Note 2736272	• SAP CRM 7.14 SP13 • SAP S/4HANA for customer management 1.0 SP04 • SAP S/4HANA for customer management 2.0 SP03
Supplier Relationship Management (SRM)	SRM-Specific KPro implementation	SAP Note 2953378	• SAP CRM 7.0 EhP3 SP19 • SAP SRM 7.0 EhP4 SP14

A Appendix

A.1 Overview of Attachment Frameworks

Generic Object Services (GOS)

In GOS, any file type can be attached. Files can only be added or deleted and are otherwise read-only. GOS also supports the creation of notes and URLs linked to the business object, as seen in figure 5. GOS documents can be archived together with the BOR object data using archiving class SGOS, however, this class is not widely used in standard ILM objects.

For more information, refer to the Generic Object Services documentation on the SAP Help Portal.

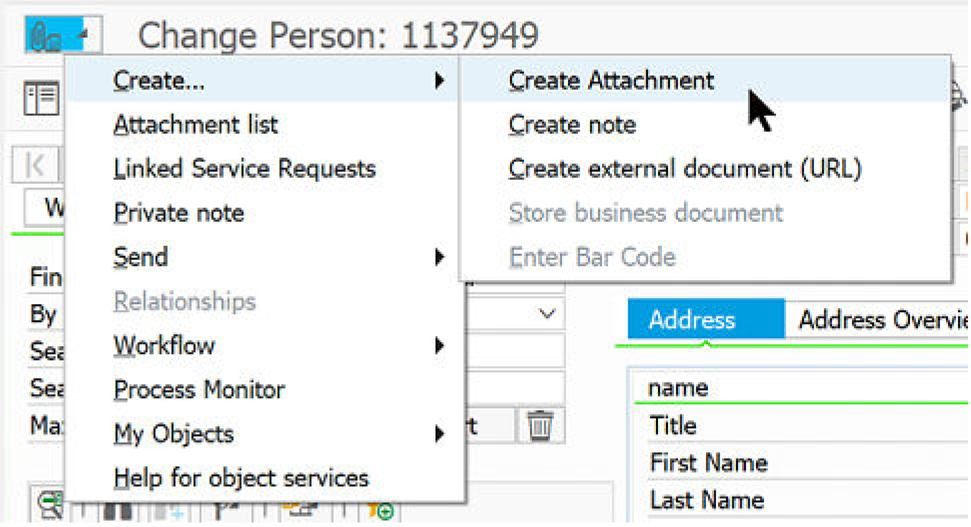


Figure 5: GOS Menu

ArchiveLink (ARL)

ARL documents can also be accessed from the same menu as shown in figure 5, if the respective BOR object was configured in transaction `OAD5`. Select the option *Store Business Document* and *Enter Barcode* to store documents using ArchiveLink.

For more information, refer to the ArchiveLink documentation on the SAP Help Portal.

Business Document Service (BDS)

Like GOS, the BDS-Framework is based on the KPro Document Management Framework. BDS offers a richer functionality for documents, such as MS Office integration, versioning, and document variants.

For more information, refer to the Business Document Service documentation on the SAP Help Portal.

A.2 Activity Diagrams: Destruction of Attachments

A.2.1 Destruction from the Archive

Generic Object Services (GOS)

As shown in figure 6, the destruction of attachments is performed in a two-step approach. For this, the two methods of *BAdI: Clean Up Activity Before Deleting an Archive File* (`BADI_ILM_PRE_DESTR_FILE_ACTION`) are used.

During the processing of the archiving files (ADK resources), BAdI method `OPERATE_ON_DESTRUCTION` is called to identify the BOR object instances. These instances are buffered in an internal table. The section colored in purple shows the determination of the BOR object instances derived from the settings made in ILM Customizing. Alternatively, the BOR keys can be determined by implementing *BAdI: Determination of BOR Object Instance Key for Destroying Attachments* (`BADI_IRM_ATTACHMENTS`).

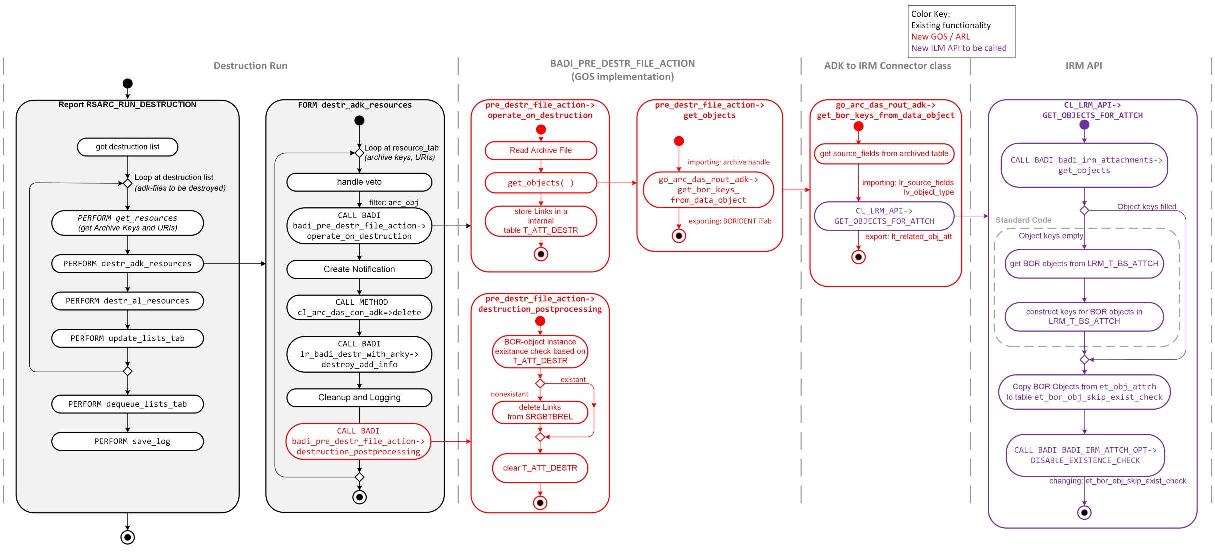

Color Key:
Existing functionality
New GOS / ARL
New ILM API to be called
Destruction Run
BADI_PRE_DESTR_FILE_ACTION (GOS implementation)
ADK to IRM Connector class
IRM API
Report RSARC_RUN_DESTRUCTION
get destruction list
Loop at destruction list (adk-files to be destroyed)
PERFORM get_resources (get Archive Keys and URIs)
PERFORM destr_adk_resources
PERFORM destr_al_resources
PERFORM update_lists_tab
PERFORM dequeue_lists_tab
PERFORM save_log
FORM destr_adk_resources
Loop at resource_tab (archive keys, URIs)
handle veto
filter: arc_obj
CALL BADI badi_pre_destr_file_action-> operate_on_destruction
Create Notification
CALL METHOD cl_arc_das_con_adk=>delete
CALL BADI lr_badi_destr_with_arky-> destroy_add_info
Cleanup and Logging
CALL BADI badi_pre_destr_file_action-> destruction_postprocessing
pre_destr_file_action-> operate_on_destruction
Read Archive File
get_objects( )
store Links in a internal table T_ATT_DESTR
pre_destr_file_action-> get_objects
importing: archive handle
go_arc_das_rout_adk-> get_bor_keys_ from_data_object
exporting: BORIDENT iTab
pre_destr_file_action-> destruction_postprocessing
BOR-object instance existance check based on T_ATT_DESTR
existant
nonexistant
delete Links from SRGBTBREL
clear T_ATT_DESTR
go_arc_das_rout_adk-> get_bor_keys_from_data_object
get source_fields from archived table
importing: lr_source_fields lv_object_type
CL_LRM_API-> GET_OBJECTS_FOR_ATTCH
export: lt_related_obj_att
CL_LRM_API-> GET_OBJECTS_FOR_ATTCH
CALL BADI badi_irm_attachments-> get_objects
Object keys filled
Standard Code
Object keys empty
get BOR objects from LRM_T_BS_ATTCH
construct keys for BOR objects in LRM_T_BS_ATTCH
Copy BOR Objects from et_obj_attch to table et_bor_obj_skip_exist_check
CALL BADI BADI_IRM_ATTCH_OPT-> DISABLE_EXISTENCE_CHECK
changing: et_bor_obj_skip_exist_check

Figure 6: ILM Destruction Run for GOS Attachments

ArchiveLink (ARL)

Figure 7 depicts the process for destroying ArchiveLink attachments. Again, standard ILM functionality is used to determine the BOR object instances. The difference to GOS is that attachment files are automatically removed from the Content Server once the last link is deleted.

The implementation for BDS documents is similar to the ARL implementation, therefore it will not be further explained at this point. The only difference is the handling of ILM references, which is specific to ARL. It skips the destruction as soon as a reference was found.

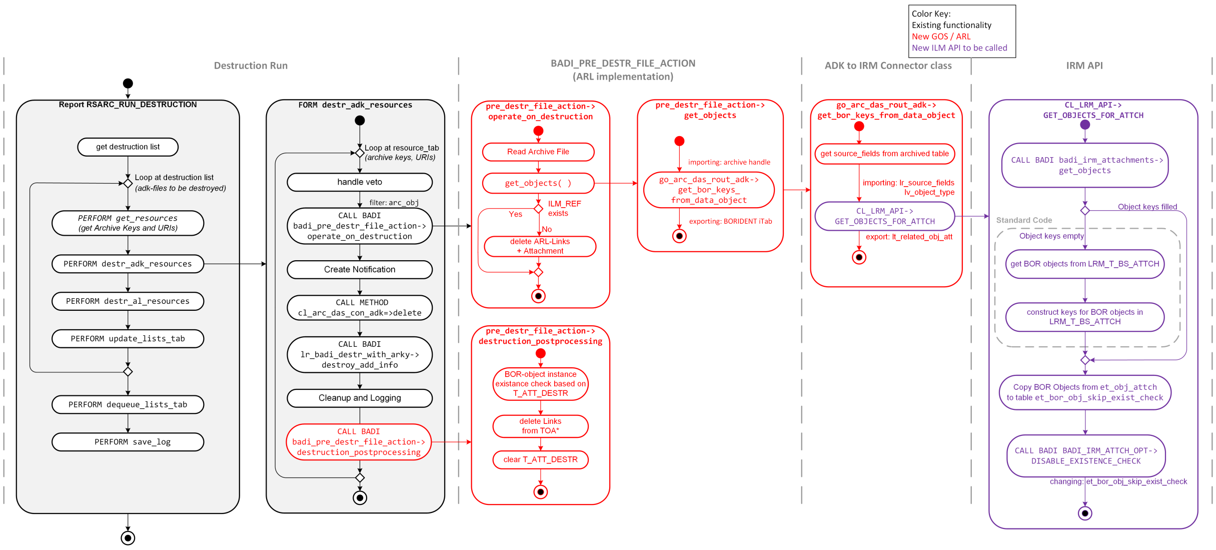


Figure 7: ILM Destruction Run for ArchiveLink Attachments

A.2.2 Destruction from the Database

Generic Object Services (GOS)

Figure 8 shows the ILM destruction process for GOS attachments. It uses implementations of *BAdI: Clean Up Activities Prior to Destroying a Data Object with Data Archiving* (BADI_ILM_PRE_DESTR_OBJ_ACTION). Similar to the destruction during archiving, the BAdI method OPERATE_ON_DESTRUCTION first collects the BOR instances to be destroyed. After successful deletion of the leading data, the second method OPERATE_ON_FILE_DESTRUCTION is called to find and destroy the attachments links to the BOR objects gathered in the first step.

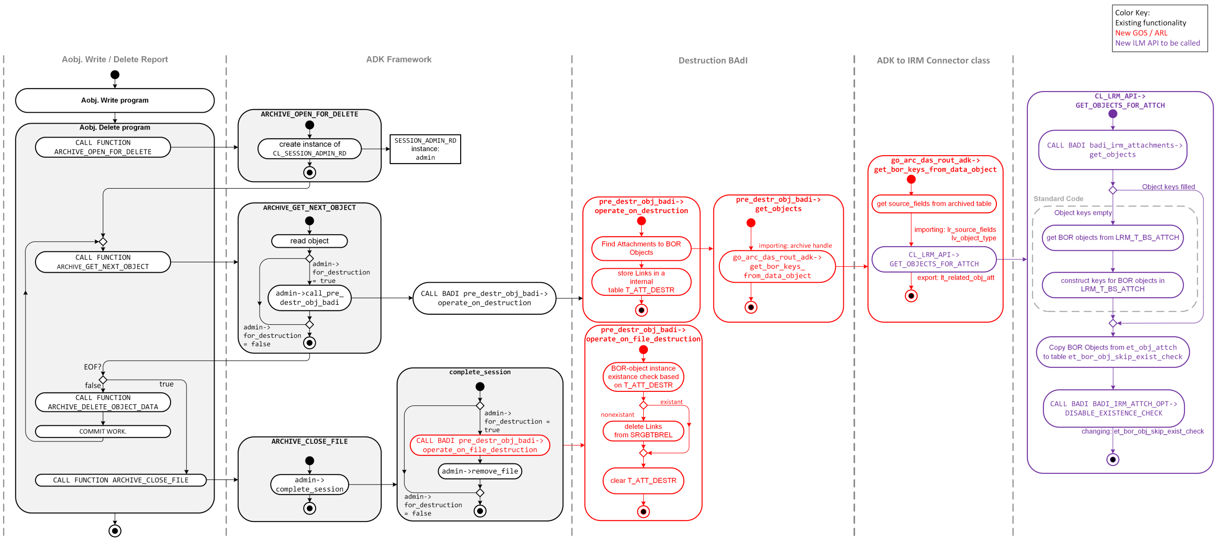

Color Key:
Existing functionality
New GOS / ARL
New ILM API to be called
Aobj. Write / Delete Report
ADK Framework
Destruction BAdI
ADK to IRM Connector class
Aobj. Write program
Aobj. Delete program
CALL FUNCTION ARCHIVE_OPEN_FOR_DELETE
CALL FUNCTION ARCHIVE_GET_NEXT_OBJECT
EOF?
false
true
CALL FUNCTION ARCHIVE_DELETE_OBJECT_DATA
COMMIT WORK.
CALL FUNCTION ARCHIVE_CLOSE_FILE
ARCHIVE_OPEN_FOR_DELETE
create instance of CL_SESSION_ADMIN_RD
SESSION_ADMIN_RD instance: admin
ARCHIVE_GET_NEXT_OBJECT
read object
admin-> for_destruction = true
admin->call_pre_ destr_obj_badi
admin-> for_destruction = false
CALL BADI pre_destr_obj_badi-> operate_on_destruction
ARCHIVE_CLOSE_FILE
admin-> complete_session
complete_session
admin-> for_destruction = true
CALL BADI pre_destr_obj_badi-> operate_on_file_destruction
admin->remove_file
admin-> for_destruction = false
pre_destr_obj_badi-> operate_on_destruction
Find Attachments to BOR Objects
store Links in a internal table T_ATT_DESTR
pre_destr_obj_badi-> operate_on_file_destruction
BOR-object instance existance check based on T_ATT_DESTR
existant
nonexistant
delete Links from SRGBTBREL
clear T_ATT_DESTR
pre_destr_obj_badi-> get_objects
importing: archive handle
go_arc_das_rout_adk-> get_bor_keys_ from_data_object
go_arc_das_rout_adk-> get_bor_keys_from_data_object
get source_fields from archived table
importing: lr_source_fields lv_object_type
CL_LRM_API-> GET_OBJECTS_FOR_ATTCH
export: lt_related_obj_att
CL_LRM_API-> GET_OBJECTS_FOR_ATTCH
CALL BADI badi_irm_attachments-> get_objects
Object keys filled
Standard Code
Object keys empty
get BOR objects from LRM_T_BS_ATTCH
construct keys for BOR objects in LRM_T_BS_ATTCH
Copy BOR Objects from et_obj_attch to table et_bor_obj_skip_exist_check
CALL BADI BADI_IRM_ATTCH_OPT-> DISABLE_EXISTENCE_CHECK
changing: et_bor_obj_skip_exist_check

Figure 8: ILM Destruction from Database with GOS Attachments

ArchiveLink (ARL)

In figure 9, the ILM destruction run is shown for ARL attachments. The same similarities and differences to GOS and BDS attachments apply as for archive destruction.

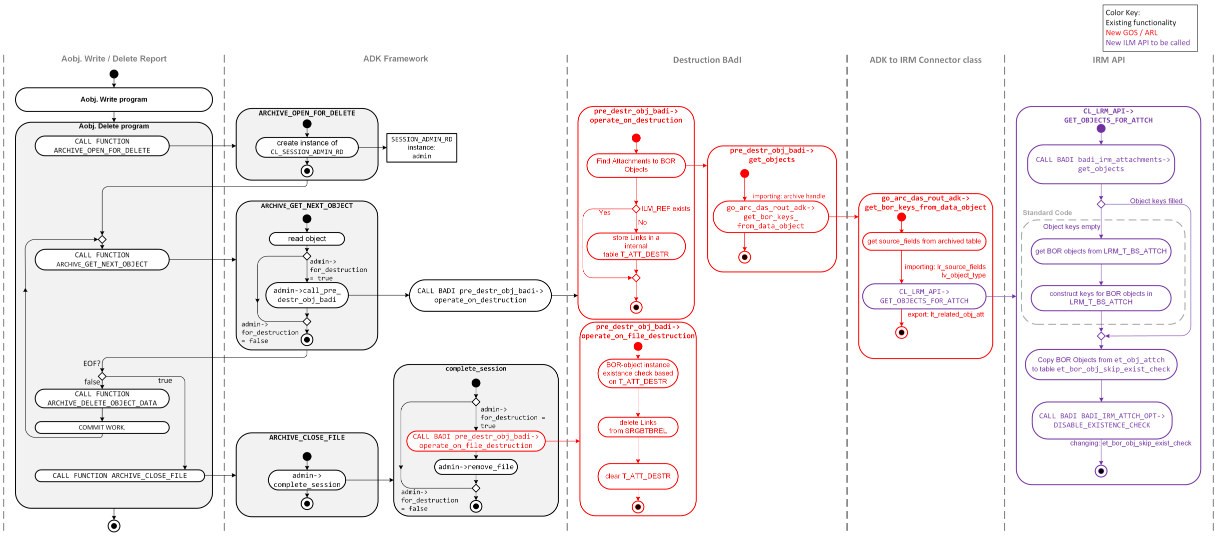

Color Key:
Existing functionality
New GOS / ARL
New ILM API to be called
Aobj. Write / Delete Report
ADK Framework
Destruction BAdI
ADK to IRM Connector class
IRM API
Aobj. Write program
Aobj. Delete program
CALL FUNCTION ARCHIVE_OPEN_FOR_DELETE
CALL FUNCTION ARCHIVE_GET_NEXT_OBJECT
EOF?
false
true
CALL FUNCTION ARCHIVE_DELETE_OBJECT_DATA
COMMIT WORK.
CALL FUNCTION ARCHIVE_CLOSE_FILE
ARCHIVE_OPEN_FOR_DELETE
create instance of CL_SESSION_ADMIN_RD
SESSION_ADMIN_RD instance: admin
ARCHIVE_GET_NEXT_OBJECT
read object
admin-> for_destruction = true
admin->call_pre_ destr_obj_badi
admin-> for_destruction = false
CALL BADI pre_destr_obj_badi-> operate_on_destruction
ARCHIVE_CLOSE_FILE
admin-> complete_session
complete_session
admin-> for_destruction = true
CALL BADI pre_destr_obj_badi-> operate_on_file_destruction
admin->remove_file
admin-> for_destruction = false
pre_destr_obj_badi-> operate_on_destruction
Find Attachments to BOR Objects
ILM_REF exists
Yes
No
store Links in a internal table T_ATT_DESTR
pre_destr_obj_badi-> operate_on_file_destruction
BOR-object instance existance check based on T_ATT_DESTR
delete Links from SRGBTBREL
clear T_ATT_DESTR
pre_destr_obj_badi-> get_objects
importing: archive handle
go_arc_das_rout_adk-> get_bor_keys_ from_data_object
go_arc_das_rout_adk-> get_bor_keys_from_data_object
get source_fields from archived table
importing: lr_source_fields lv_object_type
CL_LRM_API-> GET_OBJECTS_FOR_ATTCH
export: lt_related_obj_att
CL_LRM_API-> GET_OBJECTS_FOR_ATTCH
CALL BADI badi_irm_attachments-> get_objects
Object keys filled
Standard Code
Object keys empty
get BOR objects from LRM_T_BS_ATTCH
construct keys for BOR objects in LRM_T_BS_ATTCH
Copy BOR Objects from et_obj_attch to table et_bor_obj_skip_exist_check
CALL BADI BADI_IRM_ATTCH_OPT-> DISABLE_EXISTENCE_CHECK
changing: et_bor_obj_skip_exist_check

Figure 9: ILM Destruction from Database with ARL Attachments

Important Disclaimers and Legal Information

Hyperlinks

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

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

Videos Hosted on External Platforms

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

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

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

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

www.sap.com/contactsap

THE BEST RUN SAP