MFT Standard Solution - Overview

Version: 6.5.2 SP55

February 2, 2018

SEEBURGER
BUSINESS INTEGRATION

Table of Contents

Note: False configuration and/or improper use of communication components may result in significant charges from your telecommunication provider. Also consider configuration changes initiated by your telecommunication provider. SEEBURGER is not liable for related additional costs.

Note: Please refer to the Knowledgebase article 13379 for an important notice concerning possible incurred charges from your telecommunication provider.

Note: Export-/Import Regulations

The SEEBURGER AG is a German corporate with subsidiaries in countries worldwide. In daily trading with our customers and partners, each SEEBURGER corporate deals in compliance with the local export and import regulations of the specific country.

SEEBURGER standard software is partly based on US technology (tomcat, Red Hat JBoss, Oracle Java etc.). To be compliant with the US Export Administration Regulation and the Treasury Department regulations SEEBURGER software and/or technical data may NOT be exported /reexported, either directly or indirectly, to any destination subject to U.S. embargos or trade sanctions unless formally authorized by the U.S. Government. Note that said embargoed destinations are subject to change and the scope of what technology is included in the embargo is specific to each embargoed country. For the most current information on U.S. embargoed and sanctioned countries, please contact the U.S. Export Administration Regulations and Treasury Department regulations.

Note: We expressly declare that the document **"SEEBURGER Legal Information"** (delivered also with your BIS installation media) is part of this documentation.

1 Activities

1.1 Common Fields

This section describes some fields commonly used in many activities.

1.1.1 Queue

The *Queue* field is used in many activities. Activities using this field will refer to this section. The *Queue* field actually refers to adapter invocation i.e. the mode how an order can be forwarded from the business process to the adapter. Generally, there are three modes available, the asynchronous invocation that queues the orders until the adapter is able to handle the order, the synchronous invocation where no queue is used and the adapter order is executed synchronously inside the business process, and the Worklist Handler (WLH) mode that also queues the orders and provides some features for a more complex configuration like resource handling.

More detailed information about queues and queue configuration can be found in the Master Adapter Configuration in the Adapter Invocation chapter.

<table>
<tr><th>Field name</th><th>Value</th><th>Description</th></tr>
<tr><td>Queue</td><td></td><td>Mandatory. Editable selection field providing all queues that can handle the underlying adapter of the activity. Three modes are available in order to forward an order from a business process to an adapter: synchronous and asynchronous mode, and the WLH mode that is to be used for some special cases only.
The <i>Queue</i> supports:<ul><li>System variables, e.g. "${MY_QUEUE}".</li><li>Properties, e.g. "$[MY_QUEUE]".</li><li>XPath expressions, e.g. "QUEUE-$(seeutil:getCurrentDate('yyyyMMdd'))" or "QUEUE-$(seeutil:decimalFormat(seeutil:nextCounter('0000000000'))".</li></ul></td></tr>
<tr><td></td><td>Direct mode</td><td>Direct adapter invocation is used to forward orders from the process engine to the adapter. A JMS queue is used to queue the orders until the adapter is ready to handle them.

The direct mode supports orders retry handling in case of an adapter error. This actually depends on the adapter supporting and enabling a retry or not. It is possible to connect more than one adapter to one single queue.

Entering a queue that does not exist results in creating it automatically during the first execution of the business process.

<table><tr><th>Syntax</th><th>Description</th></tr><tr><td>async:</td><td>The default queue to&lt;<i>PortType</i>&gt; is used (e.g. toDtFTPComponent). This is the recommended usage for simple architectures, since this method does not require adapter configuration and queue definition.</td></tr><tr><td>direct:</td><td>Used instead of async: up to version 6.3.3. Can still be used</td></tr></table></td></tr>
</table>

The following table provides a short overview with the differences of the invocation:

Function	Worklist Handler	Direct Invocation	Sync Invocation
Can use resources like ISDN	X	O	O
Polling of adapter (WLH) queues	X	O	O
Complex order handling like validity and scheduling	X	O	O
Partner specific sending	X	O	O
Retry on error	X	X	O
Automatic creation of unknown queues	O	X	O

X: Feature supported
O: Feature not supported

1.1.2 Abort on error

The *Abort on error* field is used in many activities. Activities using this field will refer to this chapter. The *Abort on error* field allows configuring whether the process should be continued in case of an error or not:

Field name	Value	Description
Abort on error	True	Default. Abort the process directly after the error appeared. Resulting process status is "Stopped after error (9)". If a fault handler is defined that catches the type of error that just occured it will be called.
	False	Do not abort the process if an error occurs and continue with the next activity. Resulting process status is "Finished (5)" (unless a succeeding activity fails). Sets the result to be used by the next activity if necessary so that a smooth transition is guarantueed. The next activity gets the property LAST_STATUS='ERROR'. If a fault handler is defined that catches the type of error that just occured it will not be called.

1.1.3 Log message

The *Log message* field is used in many activities. Activities using this field will refer to this chapter.

Field name	Value	Description
Log message	*<Text>*[1]	Optional. Log message in case of activity success. The *Log message* supports: • Properties (page 392) e.g. $[TARGET_PATH]. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). System variables are not supported

Field name	Value	Description
		If *Log message* is left empty, a default log message is set by the activity. The log message is written to the process log file. This log file can be accessed at the end of the process from GUI attachment viewer. In addition, it is saved to disk as configured by system variable (page 392) ${MFT-LOG-SUB-PATH}. A reference to this file is set in message tracking (message tracking must be enabled).

1) Arbitrary text

Additional configuration:

System variable	Description
MFT-LOG-SUB-PATH	Default value "mft.log/$(seeutil:getCurrentDate('yyyyMMdd'))". Location for log files where the log messages are written to.

1.2 Process Flow Control Components

This section describes activities enabling process flow control, such as list handling, conditional branches, and process synchronization.

1.2.1 Sub-Processing and Branching

Short overview about sub-processing, branching and forking:
Similarities and differences between *scConditionalBranch* (page 14), *scConditionalFork* (page 15), and *scGoSub* (page 23) activities.

	scConditionalBranch	scConditionalFork	scGoSub
Short charact.	Branching inside main process	Launch of separate child worflow while main workflow continues with next activity	Execution of (reusable + configurable) template process between 2 main process activities
Execute sub-process	Condition yields TRUE	Condition yields TRUE	Always
Sub-entity class	MFT-BRANCH	MFT-BRANCH	MFT-TEMPLATE
Sub-entity	Only seen by parent entity	Only seen by parent entity	Seen by all entities (and templates)
Sub-process	Branch is real and complete part of main process	Sub-process is new and separate child process	Template is executed inside the main process
Scopes	1 process, 1 scope	2 processes, 2 scopes	1 process, 2 scopes
Process flow	Straightforward: 1 flow, activities of activated branch are copied into main process and executed as main process activities	Fork: 2 flows, main process launches child process and continues with next main activity	Function call: 1 flow, main process "stops" at goSub activity, template is executed, flow returns to main process and continues with next main process activity
Main process properties	Completely accessible	Passed to child process	Only passable to template process via input parameters (max. count: 10)
Main process attachment	Completely accessible	Passed to child process	Passed to template
Sub-process properties	Completely accessible, identical with main process properties	Not returned to main process	Only returnable to main process via output parameters (max. count: 10)
Sub-process attachment	Completely accessible, identical with main process attachment	Not returned to main process	Returnable to main process as process attachment via special bookmark
scConditionalExit in branch	Main process stops with SUCCESS	Child process stops with SUCCESS. No effects on main process.	Template stops and returns, main process continues with next activity.
scConditionalStop in branch	Main process stops with selected status	Child process stops with selected status. No effects on main process.	Template stops, main process stops with selected status.

##

1.2.2 Process Attachments and Bookmarking

Main characteristics:

- There is exactly 1 current process attachment. This is used by a majority of activities.
- The current process attachment can be bookmarked.
- The current process attachment can be replaced by a bookmark.

Supporting activities:

- Additional attachments can be bookmarked using scBookmark (page 6).
- scUseBookmark (page 7) makes the current process attachment from a bookmarked attachment. The previous project attachment is lost unless it is bookmarked, too.
- A new attachment can be created from existing bookmarks using acAppendBookmarks (page 52). The previous project attachment is lost if it not used in the bookmarks list or if it is bookmarked, too.

1.2.3 scBookmark

Caption: Bookmark attachment.

Throws Fault=acBookmark.

The *scBookmark* activity is a single file activity that creates a new bookmark from the current process attachment. Later on, this bookmark can be referenced via its name, for instance when using the *scUseBookmark (page 7)* activity or another activity providing bookmark access such as *acCompareLists (page 9)* .

Mode of operation:
Bookmarks the current process attachment with a unique name in order to reference it later on via this name. Bookmarks are used by the following activities: *scUseBookmark (page 7),* *acAppendBookmarks (page 52)* , *acCompareLists (page 9,)* *scForEach (page 10,)* *acTracker (page 47,)* *acConverter (page 63)* , *acComposeMIME (page 151)* , and *acPGPEncryptMD*.

Field name	Value	Description
Bookmark	*<Name>*[1]	Mandatory. Editable text field. Default value is 'ATTACHMENT_1'. Case-sensitive name used for storing and accessing the attachment. Invalid characters ", &, < cannot be entered. The *Bookmark* supports: • Properties (page 392), e.g. '$[MY_BOOKMARK_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). System variables are not supported. ⚠ Multiple bookmarks are allowed: Pay attention to unique names in order to avoid overwriting existing bookmarks by accident. Since a bookmark is actually a special kind of property, pay attention to property names, too. An existing property is overwritten if its name is used as bookmark name.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Arbitrary case-sensitive name except invalid characters: ", &, <.
2) Arbitrary text.

Preconditions:

- Available process attachment.

Result:

- New bookmark reference: property with name from *Bookmark* field value and the process attachment as value.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 'Bookmarked'.
- <BookmarkName>_HISTORY := Bookmark history string.
- Property with name from *Bookmark* and the process attachment as value.

Possible cases of failure:

Possible case of failure	Success or error behavior
Bookmark name consisting of blanks	Success. Different number of blanks lead to different bookmarks.
Already existing bookmark name	Success. Existing bookmark is overwritten.
Bookmark name already used for a property.	Success. Existing property is overwritten.

Policies: none

1.2.4 scUseBookmark

Caption: Restore attachment from bookmark.

Throws Fault=acUseBookmark.

The *acUseBookmark* activity sets the current attachment to the specified bookmark. Booksmarks can be created using the *scBookmark* (page 6) activity.

Field name	Value	Description
Bookmark	*<Name>*[1]	Mandatory. Editable text field. Default value is 'ATTACHMENT_1'. Case sensitive name of an existing bookmarked attachment. Invalid characters ", &, < cannot be entered. The *Bookmark* supports: • Properties (page 392), e.g. '$[MY_BOOKMARK_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). System variables are not supported.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Arbitrary case-sensitive name except invalid characters ", &, <.
2) Arbitrary text.

Preconditions:

- Available bookmark.

Result:

- New process attachment.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 'Used bookmark'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Bookmark name consisting of blanks	Success. The bookmark is created.
Non-existing bookmark name	Error. Throws acUseBookmark fault.

Policies: none

1.2.5 acAddressList

Caption: Get address list.

Throws Fault=acAddressList.

The *acAddressList* activity is a single file activity that allows to retrieve all available adresses of a specified communication profile and to return them as a CSV list, called ADDRESSLIST object. Can be used by the *scForEach (page 10)* activity which allows looping the address list.

Field name	Value	Description
Communication profile		Mandatory. Not editable selection field. Provides available communication profiles defining the address types to look for.
	SLMP	SLMP profile. Looks for SLMP partner addresses containing the *Add ress prefix*.
	SFTP client	SFTP client profile. Looks for SFTP addresses containing the *Addres s prefix*.
Address prefix	*<Prefix>*[1]	Mandatory. Editable text field. Empty by default. Must be part of the address names which are to be listed. The *Address prefix* supports: • Properties (page 392), e.g. $[MY_PREFIX]. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards, regular expressions and system variables are not supported.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Search string for addresses.
2) Arbitrary text.

Preconditions: none.

Result:

- New process attachment: List of found addresses in CSV format.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 100.
- LIST_COUNT := Number of addresses matching the *Address prefix*.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing property in *Address prefix*	Success. Empty address list since the property expression is not replaced and will most likely not match any address.
Invalid XPath expression in *Address prefix*	Error. Throws acAddressList fault.

Policies: none.

1.2.6 acCompareLists

Caption: CompareLists.

Throws Fault=acCompareLists.

The *acCompareLists* activity is an activity that compares 2 lists and returns the differences in between.

Mode of operation:
Compares a target list with a source list and returns the differences between them. The current process attachment contains the target list. The bookmarked attachment contains the source list. In case of FILELIST type, the source files are detected that are not yet part of the target list or that have been changed or updated. List type and comparison mode are configurable.

Field name	Value	Description
List type		Mandatory. Non-editable selection field. Provides available list types. Current process attachment and content of bookmark in *Source list* must match the selected list type. ADDRESSLIST type is not supported.
	FILELIST	Default. File list type. Expects a FILELIST object, i.e. a CSV file with file entries. A FILELIST object is created by the following activities: *acFileClientList* (page 105), *acFTPList* (page 202), *acSFTPList* (page 221), *acZIPList* (page 149), acMIMEList (page 156), *acJMSListQueue* (page 251), *acSEEFXInboxList* (page 291) and *acSEEFXViewList2* (page 300).
Source list	*<Bookmark>*[1]	Mandatory. Editable text field. Default value '<set name to bookmarked attachment>'. Case-sensitive name of existing bookmark. Expects a list with type set by *List type*. The *source list* supports: • System variables (page 392), e.g. ${MY_BOOKMARK}. • Properties (page 392), e.g. $[MY_BOOKMARK].

Field name	Value	Description
		- XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Compare method		Mandatory. Non-editable selection field. Provides available methods for comparing the lists.
	UPDATE	Default. Standard file comparison. Gets all source files that are not yet part of the target list (different name) or that have a different size and/or date due to *Ignore timestamp* switch.
Ignore timestamp		Mandatory. Additional comparison configuration switch for list entries with identical names.
	True	Comparison by list entry size only.
	False	Comparison by list entry size and date.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to bookmark.
2) Reference to master data.

Preconditions:

- Script Executor adapter started.
- Available process attachment: contains a CSV file list.
- Referenced bookmark: contains a CSV file list.

Result:

- New process attachment: contains comparison result as CSV file list.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 100.
- LIST_COUNT := Number of entries in comparison result list.

Possible cases of failure:

Possible case of failure	Success or error behavior
Source list: name of not existing bookmark	Error. Throws acCompareLists fault.
Source list: arbitrary text	Error. Throws acCompareLists fault.

Policies: none.

1.2.7 scForEach

Caption: For each element in list.

Throws Fault=acForEach.

The *scForEach* activity is an activity that allows looping a CSV list that is hold by the current process attachment. Uses the *Extract* method from the *BIS File Content Extractor* component (refer to the chapter *4.4.1 Extract Base Components* of the BIS6 manual (####) ## ## ### ### #### ######## ######### ######) ### #### #######).

Mode of operation:
For each entry in the list, the specified sub-entity is called as sub process. The sub process gets all properties of the main process and the list object as an attachment. Looping is configurable and can be restricted to a subset of the list entries. The properties and attachments that are created by the sub process are not accessible by the main process after the sub process has been finished.

Field name	Value	Description
List type		Mandatory. Non-editable selection field. Provides the available file list types. Expects the selected list object to be in the current process attachment. Each list object is represented by a special CSV format.
	FILELIST	Default value. A FILELIST object, i.e. a CSV file containing a file list. Creates a bunch of properties for each list entry that can be used by the *Gosub* sub-process activities (see below). ℹ A FILELIST object is created by the following activities: *acFileClientList* (page 105), *acFTPList* (page 202), *acSFTPList* (page 221), *acZIPList* (page 149), *acMIMEList* (page 156), *acJMSListQueue* (page 251), *acSEEFXInboxList* (page 291) and *acSEEFXViewList2* (page 300).
	ADDRESSLIST	An ADDRESSLIST object, i.e. a CSV file containing an address list. Creates a bunch of properties for each list entry that can be used by the *Gosub* sub-process activities (see below). ℹ An ADDRESSLIST object is created by the following activities: *acAddressList* (page 8).
Start with index	*<Int>*[1]	Mandatory. Editable number field. Default value is 1. The start index for looping iteration.
Maximum iterations	*<Int>*[1]	Mandatory. Editable number field. Default value is 100. Maximum number of looping iterations.
Gosub	*<Ref>*[2]	Mandatory. Non-editable selection field. Default value '<configure me>'. The sub-entity to call during each loop iteration. The called sub process gets all of the current properties from the main process. The sub process gets the list object as an attachment. If a former attachment is needed, bookmark it in the main process and refer to it via the *scUseBookmark* (page 7) activity. ⚠ After the sub process has been finished the properties and attachments it has created are not accessible any more by the main process.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Positive Integer.
2) Reference to entity.
3) Arbitrary Text.

Preconditions:

- BIC Content Extractor node adapter started.
- Available process attachment: CSV format expected.

Result:

- Current process attachment: not changed.

Sets the following properties:

- LIST_TYPE := FILELIST or ADDRESSLIST depending on *List type*.
- FOREACH_COUNT := Number of loops.
- FOREACH_STATUS := 200 success
  205 success, empty list
  415 invalid list type
  500 no sub entity available

Sets the following properties in case of errors:

- FOREACH_FAILURE_LIST := Output attachment with all failed list entries (i.e. sub-process failures for appropriate list entries). Can occur in cases where the AbortOnError flags of the sub process activities are not set.
- FOREACH_STATUS := 204, in case of failed list entries.

Sets the following properties while looping the CSV entries in case of FILELIST type:

Property name[1]	**Existing**	**File activity meaning**[2]	**JMS activity meaning**[3]
ENTRY_FILE_NAME	Always	File name	JMS message ID
ENTRY_FILE_PATH	Always	File path (may be empty)	Not available: value is 'NOT_SET'
ENTRY_FILE_SIZE	Always	File size	Not available: value is 'NOT_SET'
ENTRY_FILE_DATE	Always	File date (may be empty)	Message date
ENTRY_FILE_SUB_PATH	Not always	Sub path	Not available

1) The table shows the meanings for the properties depending on the activity that created the FILELIST.
2) File activities are the following: , , , *acMimeList* (page 156), , *acSEEFXInboxList* (page 291), *acSEEFXViewList2* (page 300). Each CSV entry represents a file.
3) JMS activities are the following: . Each CSV entry represents a message.

FILELIST CSV object

Expected element delimiter: ';' (hex: 3B).

Expected segment delimiter: 'CRLF' (hex: 0D0A).
Expected columns:

1. Row count
2. List type depending on adapter that fetched the file/message list
3. File name
4. File path, empty if not available (depending on adapter type)
5. File size, 'NOT_SET' if not available
6. File time (last modification timestamp)
7. Priority, 'NOT_SET' if not available
8. Importance, 'NOT_SET' if not available
9. Content type, 'NOT_SET' if not available
10. File sub-path, 'NOT_SET' if not available

Sets the following properties while looping the CSV entries in case of ADDRESSLIST type:

Property name[1]	Existing	Meaning[2]
ENTRY_ADDRESS_NAME	Always	Address name
ENTRY_ADDRESS_UUID	Always	Address UUID
ENTRY_CONNECTION_UUID	Not always	Connection UUID
ENTRY_PROTOCOL	Always	DT protocol, either 'SFTP' or 'SLMP'

1) The table shows the meanings for the properties depending on the activity that created the ADDRESSLIST.
2) Address activities are the following: *acAddressList* (page 8). Each CSV entry represents a master data address.

ADDRESSLIST CSV object

Expected element delimiter: ';' (hex: 3B).
Expected segment delimiter: 'CRLF' (hex: 0D0A).
Expected columns:

1. Row count
2. Protocol depending on adapter (currently SLMP or SFTP)
3. Address name
4. Address UUID
5. Connection name
6. Connection UUID

Possible cases of failure:

Possible case of failure	Success or error behavior
BIC Content Extractor node adapter not started	Error. Throws acForEach fault.
Empty CSV file	Success. FOREACH_COUNT set to 0. FOREACH_STATUS set to 205.
Invalid CSV file	Error. Throws acForEach fault.

Depending on the activities used in the sub process, errors reported by the activities do not interrupt the process. This means, the *scForEach* activity continues looping until the end of list is reached. In this case, it is possible to check the general execution state after the *scForEach* activity is finished.

Policies: none.

Example: MFTForEachHowTo (page 413).

1.2.8 scConditionalBranch

Caption: Conditional branch.

Throws Fault=acConditionalBranch.

The *acConditionalBranch* activity is an activity that allows executing a branch-process inside the parent process. Thus, the branch-process is able to access all properties of the parent process as well as its current process attachment.

Mode of operation:
First, the condition is evaluated. If the result is 'FALSE' the remaining activities of the parent process are executed. If the result is 'TRUE' the activities of the branch process replace the remaining parent process activities and are thus executed as parts of the parent process. Both branches have full access to the process properties and attachments.
Refer to *Sub-Processing and Branching* (page 4) for an overview of sub-processing flow activities.

Field name	Value	Description
Condition	*<Cond>*[1]	Mandatory. Editable text field. Default value 'true()'. Boolean XPath expression. Supports: • System variables (page 392), e.g. ${MY_XPATH_EXPR}. • Properties (page 392), e.g. $[MY_COMPARISON] or starts-with('$[FROM_PATH]', 'test') or contains('$[BOOKMARKS]', ';') or string-length('$[FROM_PATH]') > 0. • XPath expressions (page 392), resulting to boolean values, but without the $(...) notation, e.g. true() or false() or starts-with('$[FROM_PATH]', 'Cond'). I A string comparison needs quoted argument, for instance: '$[FROM_NAME]'='TEST_SYSTEM'
Branch to process		Mandatory. Non-editable selection field. Provides available sub-entities to be executed as branch-process inside the parent process if the evaluation of *Condition* yields 'True'.
	<configure me>	Default. Corresponds to an empty internal value. Can be stored, but can not be executed. When trying to execute, an exception is thrown with the message: "No branch process set in scConditionalBranch".
	<Ref>[2]	Reference to sub-process (MFT-BRANCH sub-entity).

1) Valid condition.
2) Reference to sub entity.

Preconditions:

- Current process attachment (if needed by executed branch).

Result:

- Depends on executed branch.

Sets the following properties: none.

Possible cases of failure:

Possible case of failure	Success or error behavior
Condition with not existing property	Success. Condition evaluated to 'False'.
Condition with invalid XPath expression	Error. Throws acConditionalBranch fault.
Empty *Branch to process*	Error. Throws acConditionalBranch fault.
Branching sub-entity without activities	Success.

Policies: none

1.2.9 scConditionalFork

Caption: Conditional fork.

Throws Fault=acConditionalFork.

The *acConditionalFork* activity is an activity that executes a sub-process as separated child process in both synchronous or asynchronous mode. The current process attachment is passed to the child process.

Mode of operation:
When using the synchronous *Call mode*, the *acConditionalFork* activity waits for the reply of the started child process. It fails if the child process cannot be started and gets stuck as waiting message. The asynchronous *Call mode* is faster than the synchronous one since the *acConditionalFork* activity does not wait and immediately continues. It will never know if the child process could be started successfully. Refer to *Sub-Processing and Branching* (page 4) for an overview of sub-processing flow activities.

Field name	Value	Description
Condition	*<Cond>*[1]	Mandatory. Editable text field. Default value 'true()'. Boolean XPath expression. If this expression yields 'True' the specified sub process is executed. Supports: • System variables (page 392), e.g. ${MY_XPATH_EXPR}. • Properties (page 392), e.g. $[MY_COMPARISON] or false()=starts-with('$[FROM_PATH]', 'test') or true()=contains('$[BOOKMARKS]', ';') or string-length('$[FROM_PATH]') > 0. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). A string comparison needs quoted argument, for instance: '$[FROM_NAME]'='TEST_SYSTEM'

Field name	Value	Description
Fork new process		Optional. Non-editable selection field. Provides available sub-entities to be executed as separate child process if the evaluation of *Condition* yields 'True'.
	<configure me>	Default. The entity is storable with this entry but not executable, an exception is thrown with the message: "No fork process set in scConditionalFork".
	<*Ref*>[2]	Reference to sub-process (MFT-BRANCH sub-entity).
Call mode		Optional. Editable selection field. Provides 2 modes.
	<default (sync)>	Default. Synchronous call of new process. Value 'SYNC' can be used instead. The new process is marked as child process. It is started with a synchronous MFT-ProcessEvent which returns the response properties 'TRACKID' and 'CONTEXT_ID' before continuing with the next activity. The calling process waits for the response and continues with the next activity after receiving it. I The current version only logs the response properties, but does not really take them.
	<asynchronous>	Asynchronous call of new process. Value 'ASYNC' can be used instead. The new process is not marked as child process. It is started with an asynchronous MFT-ProcessEventAsync. After starting the sub-process, the calling process immediately continues with the next activity. There is no guarantee that the new process is really started and does not get stuck as a waiting message.

1) Valid condition.
2) Reference to MFT-BRANCH sub entity.

Preconditions:

- Current process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- RESULT := Forked=true if *Condition* yields 'True' and process is forked.
  RESULT := Forked=false if *Condition* yields 'False' and process is not forked.
- _fork_counter := incremented by 1.

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
Condition with not existing property		Error. Throws acConditionalFork fault.
Condition with invalid XPath expression		Error. Throws acConditionalFork fault.
Empty or '<configure me>' *Branch to process*		Error. Throws acConditionalFork fault.
Branching sub-entity without activities		Success.
Branching sub-entity inactive		Success. Sub-process executed.
Branching sub-entity removed		Error. Throws acConditionalFork fault.
Invalid call mode		Success. Automatically switched to SYNC call mode.
Sub-process cannot be started	SYNC	Error. Throws acConditionalFork fault.
Sub-process cannot be started	ASYNC	Success. Process continues with next activity.
Sub-process fails after successful start (response)	SYNC	Success. Process continues with next activity.
Sub-process fails	ASYNC	Success. Process continues with next activity.

Policies: none

1.2.10 scConditionalStop

Caption: Conditional stop.

Throws Fault=acConditionalStop.

The *scConditionalStop* activity is an activity that can stop or interrupt the current process and set it to the specified *Interrupt state*. The simpler *scConditionalExit (page 22)* activity can stop and finish the current process with the final state "SUCCESS (9)".

Mode of operation:
Refer to **Interactions with other flow components** further below for more details.

⚠ The *scConditionalExit (page 22)* activity behaves in a different way than the *scConditionalStop* activity when used in a template process called by the *scGoSub* (page 23) activity.

Field name	Value	Description
Condition	*<Cond>*[1]	Mandatory. Editable text field. Default value 'true()'. Boolean XPath expression. The *Condition* supports: • System variables (page 392), e.g. ${MY_XPATH_EXPR}. • Properties (page 392), e.g. $[MY_COMPARISON] or false()=starts-with('$[FROM_PATH]', 'test') or true()=contains('$[BOOKMARKS]', ';') or string-length('$[FROM_PATH]') > 0.

Field name	Value	Description
		• XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ℹ A string comparison needs quoted argument, for instance: '$[FROM_NAME]'='TEST_SYSTEM'
Interrupt state		Mandatory. Not editable selection field. Provides available interrupt states corresponding to process states.
	Paused	Default. The process is stopped. Resulting process status is "Stopped (8)". The process can be continued (resuming with next activity) or terminated subsequently.
	Failed	The process is stopped with error. If configured, a fault handler is executed. Resulting process status is "Stopped after error (9)". The process can be continued (resuming with next activity), retried, or terminated subsequently.
	Finished success	The process is finished successfully. Resulting process status is "Finished (5)".
	Finished terminated	The process is terminated. Resulting process status is "Terminated (2)".
	Delay for	The process is interrupted and kept in running state, i.e. process status "Running (1)", for the amount of time specified by *Delay in minutes*. The process is continued after the specified delay is expired. Makes *Delay in minutes* visible.
	Delay until	The process is interrupted and kept in running state, i.e. process status "Running (1)", until the time of day specified by *Delay until (hh:mm)*. The process is continued after the specified delay is expired. Makes *Delay until (hh:mm)* visible.
Delay in minutes	*<Int>*[2]	Mandatory. Editable numeric text field. Empty by default. Minimum value 0. Maximum value 2.147.483.647 minutes (max. Integer (MAXINT)). Visible when *Interrupt state* is set to 'Delay for'. Value 0 means no delay. ℹ *acHoldFor* (page 41) can also be used for a delay in minutes. ⚠ If the value if greater than MAXINT the duration of the process interrupt is set to 9999 minutes. Note that no error is thrown. The process is running as usual. The $[CONDITIONAL_STOP_FOR_INFORMATION] is set which holds an appropriate information text. System variables, properties and XPath expressions are not supported.

Field name	**Value**	**Description**
Delay until (hh:mm)	*<DateFormat>*[3]	Mandatory. Editable text field. Empty by default. Visible when *Interrupt state* is set to 'Delay until'. Time of day until which the process is delayed. Current date is always used. Thus, a time specification in the past means no delay. The *Delay until (hh:mm)* supports: • System variables (page 392), e.g. ${MY_DELAY_UNTIL}. Properties are not supported.

1) Valid boolean XPath expression.
2) Non-negative Integer.
3) Time format hh:mm, denoting hours and minutes.

Preconditions: None.

Result:

- Current process attachment: not used and not changed.
- Process interrupted or stopped/finished with specified process state.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 'FINISHED_SUCCESS' if *Interrupt state* is set to 'Finished success'.
- RESULT := 'FINISHED_TERMINATED' if *Interrupt state* is set to 'Finished terminated'.
- RESULT := 'DELAYED' if *Interrrupt state* is set to 'Delayed for' or 'Delayed until'.
- RESULT := 'CONTINUE' if *Condition* is evaluated to FALSE or if Condition is evaluated to TRUE and *Interrrupt state* is set to 'Paused'.
- CONDITIONAL_STOP_DELAY_INFORMATION := 'Max value was exceeded. Set hold to 9999 minutes.', only if the duration is greater than MAXINT (2147483647).

Possible cases of failure:

Possible case of failure	**Mode**[1]	**Success or error behavior**
Condition with not existing property		Success. Condition evaluated to 'False'.
Condition with invalid XPath expression		Error. Throws acConditionalStop fault.
Delay in minutes field empty	Delay for	Error. Throws acConditionalStop fault.
Value in *Delay in minutes* greater than 2.147.483.647 (MAXINT)	Delay for	Success. Interruption for 9999 minutes.
Delay until field empty	Delay until	Error. Throws acConditionalStop fault.
Invalid time format in *Delay until*, e.g. 15:15:15	Delay until	Error. Throws acConditionalStop fault.

1) Selected value in *Interrupt state*.

Policies: none

Interactions with other flow components

scConditionalStop activity located in sub entitiy/template entity. Condition assumed to yield "TRUE". Refer to *Sub-Processing and Branching* (page 4) for an overview of sub-processing flow activities.

Interrupt state	*scForEach* (page 10)	*scConditionalBranch* (page 14)	*scConditionalFork* (page 15) [1]	*scGoSub* (page 23)
Finished success	• Stops sub process with success • Returns to ForEach loop in parent process • Continues with next loop member	• Stops process with "Finished (5)"	• Stops child process with "Finished (5)" • Main process is not affected	• Stops process with "Finished (5)"
Finished terminated	• Stops sub process • Returns to ForEach loop in parent process • Exits loop • Continues with next parent process activity	• Stops process with "Terminated (2)"	• Stops child process with "Terminated (2)" • Main process is not affected	• Stops process with "Terminated (2)"
Failed	• Stops sub process with error code • Stops process with "Stopped after error (9)"	• Stops process with "Stopped after error (9)"	• Stops child process with "Stopped after error (9)" • Main process is not affected	• Stops process with "Stopped after error (9)"
Paused	• Interrupts sub process with "Stopped (8)" • Process can be continued (normal process resumption with loop) or terminated with "Terminated by user (3)" (no return to loop, immediate stop at *scConditionalStop* activity)	• Interrupts process with "Stopped (8)" • Process can be continued with next activity or terminated with "Terminated by user (3)"	• Interrupts child process with "Stopped (8)" • Main process is not affected	• Interrupts template process stops with "Stopped (8)" • Process can be continued with next activity or terminated with "Terminated by user (3)" (immediate stop at *scConditionalStop* activity)
Delayed for/until	• Interrupts sub process with "Running (1)" • Process can be stopped with "Stopped (8)" or terminated "Terminated by user (3)" or continued after delay expiration (normal process resumption)	• Interrupts process with "Running (1)" • Process can be stopped with "Stopped (8)" or terminated with "Terminated by user (3)" or continued after delay expiration (normal process resumption)	• Interrupts child process with *"Running (1)"* • Main process is not affected • Child process can be stopped with "Stopped (8)" or terminated with "Terminated by user (3)" or continued after delay expiration (normal process resumption)	• Interrupts template process with "Running (1)" • Process can be stopped with "Stopped (8)" or terminated with "Terminated by user (3)" or continued after delay expiration with the next activity

1) Synchronous or asynchronous fork mode does not matter.

1.2.11 scConditionalExit

Caption: Conditional exit.

Throws Fault=acConditionalExit.

The *scConditionalExit* activity is an activity that can stop the current process with final process state "SUCCESS (9)". It represents a special case from the *scConditionalStop* (page 17) activity in case of "straightforward" processes. If used in a template called by the *scGoSub* (page 23) activity the templated process is stopped and the parent process is continued.

Mode of operation:
Refer to **Interactions with other flow components** further below for more details.

Field name	Value	Description
Condition	*<Cond>*[1]	Mandatory. Editable text field. Default value 'true()'. Boolean XPath expression. The *Condition* supports: • System variables (page 392), e.g. ${MY_XPATH_EXPR}. • Properties (page 392), e.g. $[MY_COMPARISON] or false()=starts-with('$[FROM_PATH]', 'test') or true()=contains('$[BOOKMARKS]', ';') or string-length('$[FROM_PATH]') > 0. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ℹ A string comparison needs quoted argument, for instance: '$[FROM_NAME]'='TEST_SYSTEM'

1) Valid boolean XPath expression.

Preconditions: None.

Result:

- Current process attachment: not changed.
- Process finished with final process state "SUCCESS (9)"

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 'CONTINUE' if *Condition* is evaluated to FALSE.
- RESULT := 'FINISHED_RETURN' if *Condition* is evaluated to TRUE.
- CONDITIONAL_EXIT_STATUS := value of RESULT above.

Possible cases of failure:

Possible case of failure	Success or error behavior
Condition with not existing property	Success. Condition evaluated to 'False'. Process is continued.
Condition with invalid XPath expression	Error. Throws acConditionalExit fault.

Policies: none

Interactions with other flow components

scConditionalExit activity located in sub entitiy/template entity. Condition assumed to yield "TRUE". Refer to *Sub-Processing and Branching* (page 4) for an overview of sub-processing flow activities.

scForEach **(page 10)**	***scConditionalBranch*** **(page 14)**	***scConditionalFork*** **(page 15)** [1]	***scGoSub*** **(page 23)**
• Stops sub process with success • Returns to ForEach loop in parent process • Continues with next loop member	• Stops process with "Finished (5)"	• Stops child process with "Finished (5)" • Main process is not affected	• Stops template process with success • Returns to main process • Continues with next main process activity

1) Synchronous or asynchronous fork mode does not matter.

1.2.12 scGoSub

Caption: Sub-process.

Throws Fault=acGoSub.

The *acGoSub* activity is an activity that calls a sub process and returns to the main process. The sub process is a template that provides a specific interface with input and output parameters.
Refer to *Sub-Processing and Branching* (page 4) for an overview of sub-processing flow activities.

Field name	**Value**	**Description**
Sub process	*<TemplateName>*[1]	Optional. Non-editable selection field. Default value '<configure me>'. Available entity templates (MFT-TEMPLATE).
Form		Optional. Non-editable selection field. The names of available forms for input and output fields of the template. Additional forms can be added. Refer to **Parameter Form** below for more details about forms. The templates that are available are looking quite the same: each one is set to one of these available forms. It it strongly recommended to use corresponding forms for both the template and the *scGoSub* activity. Refer to **Template Implementation** (page 32) in the sub chapter below for more details.
	Default	Default. Generic form providing up to 10 input parameters and output parameters.
	Archiving	Form providing 2 input parameters for file path and file name and 1 status output parameter.
	StoreToFileShare	Form providing 2 input parameters for folder and file name. No output parameters.

Field name	Value	Description
	<FormName>[2]	Additional custom forms if existing.
...	...	Optional. Input parameter values for fields specified by the selected *Form*. Each field type can be used.Each value is passed to the template interface as string.
...	...	Optional. Output parameter keys specified by the selected *Form*. Can be missing. Each field is an editable text field. The appropriate template output parameter value is mapped to the entered parameter key, e.g. 'OUTPUT#1' resulting in a property. Succeeding activities can access this property using '$[OUTPUT#1]'. Can support: • System variables, e.g. '${INPUT1}'. • Properties, e.g. '$[INPUT1]'. • XPath expressions, e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid name of existing and active template (MFT-TEMPLATE entity).
2) Unique form display name extracted from form xml file name.
3) Arbitrary text.

Default Form

Field name	Value	Description
Form	Default	Generic form providing up to 10 input parameters and output parameters.
Input 1	*<Text>*[1]	Optional. Editable text field. Empty by default. Input parameter value. Supports: - System variables, e.g. '${INPUT1}'. - Properties, e.g. '$[INPUT1]'. - XPath expressions, e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible if *Form* is set to 'Default'. Makes *Input 2* visible if not empty. Each *Input<n>* field launches the display of an additional field *Input<n+1>* up to the maximum number of 10 fields.
Input 2..10	*<Text>*[1]	Additional input parameter values. Each field is displayed if its predecessor is not empty.
Output 1	*<Name>*[2]	Optional. Editable text field. Empty by default. Output parameter key. Appropriate template output parameter value is mapped to this key, e.g. 'OUTPUT1'. Succeeding activities can access this property '$[OUTPUT1]'. Supports: - System variables, e.g. '${INPUT1}'. - Properties, e.g. '$[INPUT1]'. - XPath expressions, e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible if *Form* is set to 'Default'. Makes *Output 2* visible if not empty. Each *Output<n>* field launches the display of an additional field *Output<n+1>* up to the maximum number of 10 fields.
Output 2..10	*<Name>*[2]	Additional output parameter keys. Each field is displayed if its predecessor is not empty.

1) Arbitrary text.
2) Valid property name.

Archiving Form

Field name	Value	Description
Form	Archiving	Specific form providing 2 input parameters and 1 status output parameter.
Archive path	*<Text>*[1]	Optional. Editable text field. Empty by default. Input parameter value. Supports: • System variables, e.g. '${ARC_PATH}'. • Properties, e.g. '$[ARC_PATH]'. • XPath expressions, e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible if *Form* is set to 'Archiving'.
File name	*<Text>*[1]	Optional. Editable text field. Empty by default. Input parameter value. Supports: • System variables, e.g. '${FILENAME}'. • Properties, e.g. '$[FILENAME]'. • XPath expressions, e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible if *Form* is set to 'Archiving'.
Status	*<Name>*[2]	Optional. Editable text field. Empty by default. Output parameter key. Appropriate template output parameter value is mapped to this key, e.g. 'OUT_STATUS'. Succeeding activities can access this property '$[OUT_STATUS]'. Supports: • System variables, e.g. '${OUT_STATUS_KEY}'. • Properties, e.g. '$[OUT_STATUS_KEY]'. • XPath expressions, e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible if *Form* is set to 'Archiving'.

1) Text matching the needs of the called template.
2) Valid property name.

StoreToFileShare Form

Field name	Value	Description
Form	StoreToFileShare	Specific form providing 2 input parameters only.
Folder	*<Text>*[1]	Optional. Editable text field. Empty by default. Input parameter value. Supports: • System variables, e.g. '${ARC_PATH}'. • Properties, e.g. '$[ARC_PATH]'. • XPath expressions, e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible if *Form* is set to 'StoreToFileShare'.
File name	*<Text>*[1]	Optional. Editable text field. Empty by default. Input parameter value. Supports: • System variables, e.g. '${FILENAME}'. • Properties, e.g. '$[FILENAME]'. • XPath expressions, e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible if *Form* is set to 'StoreToFileShare'.

1) Text matching the needs of the called template.
2) Valid property name.

Preconditions:

- No special preconditions. Refer to the called template for preconditions to consider.

Result:

- New process attachment: only if the template provides a bookmarked attachment 'BOOKMARK_PAYLOAD'.

Sets the following properties:

- Output parameter keys as specified by the selected form.

Possible cases of failure:

Possible case of failure	Success or error behavior
Property not existing	Error. Throws acGoSub fault.
System variable not existing	Error. Throws acGoSub fault.
Sub process missing	Error. Throws acGoSub fault. ERROR_CODE := 'GOSUB_INVALID_NO_SUB_PROCESS'.
Error in sub process	Error. Throws acGoSub fault.

Policies: none

Parameter Form

The form template is based on a `masterdataform` tag. It contains components declaring input parameter values. It can contain components declaring output parameter keys. It always needs a `dependency` part connecting the parameter components to the basic *Form* selection field which is called `cobFormular` in the basic form xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<masterdataform
    alignment="vertical"
    cached="false">
    <nlsresource name="com.seeburger.customization.mft.entitytypes" />

    <!-- input parameter value fields: min 1, max 10  -->

    <!-- output parameter key fields: min 0, max 10  -->

    <dependency
        owner="cobFormular"
        <!-- upper case version of display name (see below) -->
        value="ARCHIVING">
        <!-- exactly one entry for each parameter field above, replace "xxx" by the field
 ID -->
        <element ref="xxx" />
    </dependency>

</masterdataform>
```

Input parameter value fields:

- Text fields, file fields, comboboxes, checkboxes, and numeric fields can be used.
- Up to 10 fields can be added. More than 10 fields are not supported.
- Each field must have one of 10 specified datasources. The datasource IDs are the following: "IN_1", "IN_2", ..., "IN_10".

Output parameter key fields:

- Only text fields can be used. The field must be able to be edited with a property key name.
- Up to 10 fields can be added. More than 10 fields are not supported.
- Each field must have one of 10 specified datasources. The datasource IDs are the following: "OUT_1", "OUT_2", ..., "OUT_10".

⚠ The IDs of the parameter fields must be unique for all templates. Therefore, it is recommended to use the form name as a part. If an ID is reused - most probably by accident - a duplicate ID error message occurs when opening the entity in the entity explorer.

Form file name:

- Name must match the pattern "frm<FormID>.xml". Files whose name do not match will be ignored during form import (see below). Example: "frmArchiving.xml".
- The "<FormID>" is the caseinsensitive form name and must be unique.
- The display name (used in the *Form* selection field) is created from the form name by cutting the first 3 characters and the extension: "<FormID>". Example: "Archiving".
- The internal combobox value is the upper case version of the form name. This value is needed in the dependency part. Example: "ARCHIVING".

Form file import:

1. Copy the file to "<BIS>\data\webdav\repository\conf\forms\<LS>\tpm\goSub\variants" where "<BIS>" is the BIS installation root diretory and "<LS>" the logical system.
2. Go to "<BIS>\software\solutions".
3. Execute "runConsole.bat".
4. Enter the logical system from item 1.
5. Enter the command: "mergeForms".
6. Enter "exit" to leave the console.
7. Open the entity and have a look at the drop-down list of the *Form* selection field. A new entry "<FormID>" should be displayed.

Example: "frmArchiving.xml" leading to *Form* entry "Archiving" with internal value "ARCHIVING".

```
<?xml version="1.0" encoding="utf-8"?>
<masterdataform
    alignment="vertical"
    cached="false">
    <nlsresource name="com.seeburger.customization.mft.entitytypes" />

    <textfield
        id="fileARCHIVING_IN1"
        datasource="IN_1"
        allows-system-vars="true"
        searchable="true"
        maxlength="255"
        required="false"
        numeric="false"
        uniqueconstraint="false">
        <label text="Archive path" />
    </textfield>
    <textfield
        id="tfARCHIVING_IN2"
        datasource="IN_2"
        allows-system-vars="true"
        searchable="true"
        maxlength="255"
        required="false"
        numeric="false"
        uniqueconstraint="false">
        <label text="File name" />
    </textfield>
      <textfield
        id="tfARCHIVING_OUT1"
        allows-system-vars="true"
        datasource="OUT_1"
        searchable="true"
        maxlength="255"
        required="false"
        default=""
        numeric="false"
        uniqueconstraint="false">
        <label text="Status" />
    </textfield>
    <dependency
        owner="cobFormular"
        value="ARCHIVING">
        <element ref="fileARCHIVING_IN1" />
        <element ref="tfARCHIVING_IN2" />
        <element ref="tfARCHIVING_OUT1" />
    </dependency>

</masterdataform>
```

##

1.2.12.1 Template Implementation

A template is a sub-process with a specific *Interface* component at its beginning declaring input and output parameters.

The Interface Component

Caption: Interface.

The *Interface* component specifies the property keys for the input and output parameters of the template.

Field name	Value	Description
Form		Optional. Non-editable selection field. The names of available forms for input and output parameters. Additional forms can be added (see below). The input and output parameters of the template and the calling *scGoSub* activity must match (see below).
	Default	Default. Generic form providing up to 10 input parameters and output parameters.
	Archiving	Form providing 2 input parameters for file path and file name and 1 status output parameter.
	StoreToFileShare	Form providing 2 input parameters for folder and file name. No output parameters.

Field name	Value	Description
	<FormName>[3]	Additional custom forms if existing.
<InputFields>[1]	*<PropName>*[4]	Optional. Editable text field. Empty by default. Input parameter key. The input parameter value is mapped to this key, e.g. 'INPUT1'. Succeeding activities can access this property '$[INPUT1]'. Supports: - Initialization, for instance 'INPUT1=000' initializes the input parameter key 'INPUT1' with the string value '000'. Used in case there is no parameter value passed. System variables, properties and XPath expressions are not supported. ⚠ If the same form is used for both scGoSub and Interface and system variable resolving is switched on for the scGoSub part, note that system variable are not supported in the Interface although there is a drop-down displayed. If this is to be avoided you have to use 2 forms, refer to **Parameter Form** below for more details.
<OutputField>[2]	*<Property>*[5]	Optional. Editable text field. Empty by default. Designated output parameter property key. Since this property does not yet exist at the template initiation it has to be entered as a property, e.g. '$[OUTPUT1]'. Succeeding activities should a value to this property key, e.g. *acSetProperties* (page 44) can assign a value using 'OUTPUT1=whatevervalue'. Supports: - System variables, e.g. '${OUTPUT_PARAMKEY1}'. - Properties, e.g. '$[OUTPUT1]'. - XPath expressions, e.g. $(seeutil:getCurrentDate('yyyyMMdd')).

1) One or several input fields from selected *Form*.
2) Output fields from selected *Form*. Can be missing.
3) Unique form display name extracted from form xml file name.
4) Valid property key.
5) Valid but not yet existing property in property notation $[...].

Parameter Form

The form template is quite similar to the template of the *scGoSub* (page 23) activity (refer to **Parameter Form** section in parent chapter for all details). The only difference to consider is the following:
Because the input parameter fields are in fact input parameter key fields where property keys are to be entered, only text fields can be used. If the parameter form for the *scGoSub* activity contains text fields only this form can be reused for the *Interface* component. If the parameter form for the *scGoSub* activity also contains other components a new parameter form must be created for being used by the *Interface* component. Create an Interface parameter form from the *scGoSub* parameter form replacing all components by text fields and modify the field IDs to keep them unique. Assign meaningful names to both form files indicating that they belong together as well as indicating which one is the *scGoSub* form and which one is the *Interface* form.

The Template

Entity node hierarchy:

- Root node: MFT-TEMPLATEMANAGEMENT
- Grouping sub node: MFT-TEMPLATEGROUP
- Template sub node : MFT-TEMPLATE

Important features:

- All input parameter values are accessible via the Interface input parameter key: "$[<InputParamKey>]".
- An output parameter value is not accessible until the first assignment to the output parameter property key. The assignment can be done by using the *acSetProperties* (page 44) activity.
- An attachment can be returned to the calling process by bookmarking (*scBookmark* (page 6)) it using the predefined name "BOOKMARK_PAYLOAD".
- A template can call another template using the *scGoSub* (page 23) activity. Nesting templates in the entity explorer is not possible, i.e. a template can not have a "sub-template".
- ⚠ Be aware that it is possible to create (self-)recursive or cyclic process calls. Generally, it is not recommended that a templates calls itself. If nesting several template calls be sure not to produce a cycle.
  In BIS versions 3.6.2 SP42 and higher, the 11th recyclic call is stopped with an error. In previous versions, this leaded to an OutOfMemoryException.
- ⚠ When having kind of chain calls of templates where one is calling the other the deepest nesting level supported is 10. The 11th call is topped with an error.
- A template can have sub-processes (i.e. MFT-BRANCH sub-entities). It can use all activities that call those sub-processes such as *scForEach* (page 10), *scConditionalBranch* (page 14), *scConditionalFork* (page 15).
- A template can call (sub-)sequence-flows by using the *scSequence* (page 36) activities.
- A template can be aborted by using the *scConditionalExit* (page 22) activity. The process flow returns to the main process and continues with the next activity (the same as in the non-aborted case).
  Note that using the *scConditionalStop* (page 17) activity with final states "Failed", "Finished success", and "Finished terminated" aborts both the template and the main processes.
- A template can not have a fault handler of its own. The calling process must manage fault handling by itself.
- All activities can be used just as usual when creating standard MFT processes.

Example template using the "frmArchiving.xml" form:

Purpose: Store the passed process attachment to the specified archive path.

Brief description: Check if the passed process attachment is a duplicate of an already archived attachment. If so, abort the template processing and return to the main process. If not store the attachement and return a success report to the main process.

Parameters:

- Archive path: the file system target path of the archive.
- File name: the name for the target file.
- Status: the resulting status, "DUPLICATION_DETECTED" in case the attachment is a duplicate, "ARCHIVED" in case it is not and is archived successfully.

Interface	Form	Archiving	Select the form to use for the template.
	Archive path	ARCHIVE_PATH	Enter the property key for this parameter.
	File name	FILE_NAME	Enter the property key for this parameter.
	Status	$[ARC_STATUS]	Enter the property for this parameter.
acCalculateHash	Hash type	MD5	Configure hash value calculation for the current process attachment.
acSetProperties	Key value pair 1	ARC_STATUS=DUPLICATION_DETECTED	Initialize the output parameter property with respect to conditional exit.
acDuplicateCheck	Check if the current process attachment is a duplicate of an already archived one.		
	Duplicate check ID	$[HASH_VALUE]	Enter the hash value calculated above.
	Duplication check mode	CHECK	Configure the duplicate check.
	Handle duplicate	Report duplicate	
	Duplicate check type	Duplication hash	
scConditionalExit	Condition	'$[DUPLICATE_DETECTED]'='YES'	Exit the template if a duplicate has been detected and continue in the main process.
mdStoreFile	Store the current process attachment.		
	Path	$[ARCHIVE_PATH]	The location of the archive directory.
	File name	$[FILE_NAME]	The file name to use for the attachment.
	Append mode	Overwrite	Configure the actual storing.
	Abort on error	True	
acPickFile	Pick the report file that is to be returned to the calling process (just to give an example how to return an attachment to the main process).		

	Source path	D:\Temp\Test\GoSub	
	File name	ArchiverReport.txt	
	Delete picked file	False	
	Abort on error	True	
scBookmark	Bookmark the loaded report attachment.		
	Bookmark	BOOKMARK_PAYLOAD	This bookmark name assures that the bookmarked attachment is returned to the calling process as its current process attachment.
acSetProperties	Key value pair 1	ARC_STATUS=ARCHIVED	Assign the final value to the output parameter. The calling process will see a property $[ARC_STATUS] with value 'ARCHIVED' when returning.

#

1.2.13 scSequence

Caption: Call sequential flow.

Throws Fault=acSequence.

The *acSequence* activity is an activity that can be used to implement in-sequence processing. Refer to In-Sequence Processing (page 400) for more details.

Field name	Value	Description
Profile	*<Ref>*[1]	Mandatory. Editable selection field. Master data Sequence profiles. Represents sequence configuration features timeout handling, gap handling, out-of-sequence handling, instance-down handling and reorganization (refer to the Sequence profiles section of In-Sequence Processing (page 400) for more details), i.e. critical section access and process control. The *Profile* supports: • System variables (page 392), e.g. '${SEQ_PROFILE}'. • Properties (page 392), e.g. '$[SEQ_PROFILE]'. • XPath expressions (page 392), e.g. $(upper-case('profilename')).
Name of sequence	*<SeqName>*[2]	Mandatory. Editable text field. Empty by default. The name of the sequence to be used for sequencing. If there is a sequence with this name this sequence is used. Otherwise, a new sequence is created with this name, using the sequence configuration from *Profile*, and starting with the actual sequence number value from *Sequence number* as expected sequence number. The *Name of sequence* supports: • System variables (page 392), e.g. '${SEQ_NAME}'. • Properties (page 392), e.g. '$[SEQ_NAME]' or 'Hotfolder_Receive_$[FROM_UUID]'. • XPath expressions (page 392), e.g. $(upper-case('seqname')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Special XPath expression in case of DT adapters that support in-sequence processing: $(see:getContextProperty('SEQUENCE_NA retrieves the name of the sequence counter used by the adapter.
Sequence number	*<SeqNum>*[3]	Mandatory. Editable text field. Empty by default. The current sequence number of this process. In case of JMSEvents and HTTPServerEvents, the sequence number has been set by the adapter. It is retrieved using the following XPath expression:

1) Reference to master data.
2) Arbitrary name.
3) Integer number.
4) Reference to sequence flow.
5) Arbitrary text.

Preconditions:

- Current process attachment.
- User rights for editing GUI master data: 'SHOW_SEQUENCE' and 'SEQUENCE_EDIT_ALLOWED'.

Result:

- New process attachment: if changed by sequence flow.

Sets the following properties:

- Properties set by activities in sequence flow.

Possible cases of failure:

Possible case of failure	Success or error behavior
Non-existing profile	Error. Throws acSequence fault.
Invalid property in *Sequence profile*	Error. Throws acSequence fault.
Invalid system variable in *Sequence profile*	Error. Throws acSequence fault.
No sequence with this *Sequence name*	Success. Sequence with this sequence name is created.
Invalid property in *Sequence name*	Success. Sequence with this sequence name is created.
Invalid system variable in *Sequence name*	Error. Throws acSequence fault.
Already processed *Sequence number*	Error. Throws acSequence fault.
Invalid *Sequence number* (not a number)	Error. Throws acSequence fault.
Invalid property in *Sequence number*	Error. Throws acSequence fault.
Invalid system variable in *Sequence number*	Error. Throws acSequence fault.
Non-existing *Sequence flow*	Error. Throws acSequence fault.
Inactive *Sequence flow*	Success. Works like an active sequence flow.

Sequence profile related cases of failure:

Possible case of failure	Mode	Success or error behavior
Process runtime in critical section greater than CS timeout	Keep lock	Error. Throws acSequence fault. Critical section is still locked. No waiting process is allowed to enter.

Possible case of failure	Mode	Success or error behavior
	Release lock	Error. Throws acSequence fault. Critical section is released. The next waiting process is allowed to enter.
Element arrives with sequence number exceeding allowed gap: `CurSeqNo > ExpSeqNo + MaxGap`		Error. Throws acSequence fault.
Out-of-sequence element arrives: `CurSeqNo < ExpSeqNo`	Abort process with error	Error. Throws acSequence fault.
	Process as next	Success. Process enters the critical section.

Policies: none.

Examples: *How To In-Sequence Processing* (page 421).

1.2.14 Process to Process Correlation

This section describes activities that synchronize processes on a BIS system that are independent from one another.

1.2.14.1 acSendEvent

Caption: SendEvent.

Throws Fault=acSendEvent.

The *acSendEvent* activity is a process synchronization activity that sends a message to the correlator. Some other process with an *acWaitForEvent* (page 40) activity, already started and waiting or even not yet started, is looking for this message.

Mode of operation:
Creates a message containing the current process attachment, if specified by *Send current attachment*, and the properties from the *Properties* list. This message is sent to the correlator in an asynchronous way. It is put into the Waiting messages with PPO="sendEvent/ProcessCorrelationPortType/correlate" if there is not yet a process waiting for it. The *acSendEvent* activity does not wait for a response.

Field name	Value	Description
Correlation ID	<ID>	Mandatory. Default value 'NOT_SET'. Editable text field. The *Correlation ID* supports: • Properties (page 392), e.g. '$[MY_CORRELATION_ID]'. • XPath expressions (page 392), e.g. $(upper-case('correlation')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). System variables are not supported.
Properties (key = value;...)	*<KVList>*[1]	Optional. Editable text field. List of properties to be added to the message. The *Properties (key = value;...)* supports: • Properties (page 392), e.g. '$[PROP1]=ABC;$[PROP2]=XYZ;'.

Field name	Value	Description
		• XPath expressions (page 392), e.g. $(upper-case('correlation')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Use self-defined properties only and check the names carefully because existing properties and also system properties are overwritten on the target process without any warnings. Recommendation: Use prefixes for property names. Recommendation: Never overwrite system properties. System variables are not supported.
Send current attachment	True	The current process attachment is added to the message.
	False	Default. The current process attachment is not added to the message.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Semicolon separated key-value list.
2) Arbitrary text.

Preconditions: none.

Result:

- Current process attachment: not changed.

Sets the following properties:

- CORRELATION_ID := the specified correlation ID.
- The properties from *Properties (key = value;...)* .
- sStatus := 'SUCCESS'.
- sResult := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid list syntax i.e. missing '='	Error. Throws acSendEvent fault.
Property not existing	Success. All properties are created.
Wrong syntax for XPath expression	Success. All properties are created.

Policies: none.

1.2.14.2 acWaitForEvent

Caption: Wait for event.

Throws Fault=acWaitForEvent.

The *acWaitForEvent* activity is a process synchronization activity that looks for a message matching the specified *Correlation ID*. If there is no such waiting message it waits the specified amount of time. These waiting messages can be created using the *acSendEvent* (page 39) activity.

Mode of operation:
The *acWaitForEvent* activity waits for amount of time specified by *Timeout in minutes* for a waiting message that matches the *Correlation ID*. The waiting message can carry an attachment as well as a bunch of properties which are taken by the *acWaitForEvent* activity.

Field name	**Value**	**Description**
Correlation ID	<ID>	Mandatory. Default value 'NOT_SET'. Editable text field. ID for message matching. The *Correlation ID* supports: • System variables (page 392), e.g. '${CORRELATION_ID}'. • Properties (page 392), e.g. '$[CORRELATION_ID]'.
Timeout in minutes	*<Int>*[1]	Mandatory. Default value 5 (minutes). Editable number field. The amount of time the *acWaitForEvent* is waiting for a message matching the *Correlation ID*. Properties, system variables and XPath properties are not supported.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Integer number.
2) Arbitrary text.

Preconditions:

- System variable ${MFT-MTLOGGER-MAPPING} existing.

Result:

- CORRELATION_STATUS := 'CORRELATED' in case of success.
  'TIMEOUT' in case of error/timeout.
- RESULT := 'Correlated'.
- STATUS := 'SUCCESS'.

Sets the following properties: none.

Possible cases of failure:

Possible case of failure	**Success or error behavior**
Invalid system variable in *Correlation ID*	Error. Throws acWaitForEvent fault.
Invalid property in *Correlation ID*	Error. Throws acWaitForEvent fault.
No matching message found while waiting	Success. Timeout is logged.

Policies: none.

1.2.15 acHoldFor

Caption: Hold for.

Throws Fault=acHoldFor.

The *acHoldFor* activity is an activity that allows interrupting a running process for the specified amount of time.

scConditionalStop (page 17) can also be used for the same purpose when setting the *Interrupt state* to 'Delay for'.

Field name	Value	Description
Support system variables	True	Default. Enables entering properties and system variables into *Hold for minutes*. Makes the text field visible.
	False	Configures *Hold for minutes* as numeric field. Makes the numeric field visible.
Hold for minutes	*<Int>*[1]	Mandatory. Editable **numeric field**. Default value 5 minutes. Minimum value 1 minute. Maximum value 9999 minutes. The process is interrupted and kept in running state, i.e. process status "Running (1)", for the amount of time specified . Properties, system variables and XPath expressions are not supported. Visible if *Support system variables* is switched off.
Hold for minutes	*<Int>*[1]	Mandatory. Editable **text field**. Empty by default. Minimum value 0 minutes. Maximum value 2.147.483.647 minutes (max. Integer (MAXINT)). Supports: • System variables, e.g. '${DURATION}'. • Properties, e.g. '$[DURATION]'. • XPath expressions, e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ If the value if greater than MAXINT the duration of the process interrupt is set to 9999 minutes. Note that no error is thrown. The process is running as usual. The $[HOLD_FOR_INFORMATION] is set which holds an appropriate information text. Visible if *Support system variables* is switched on.

1) Positive Integer.

Preconditions: none.

Result:

- Current process attachment: not changed.

Sets the following properties:

- HOLD_FOR_TIME := *Content of Hold for minutes*.
- HOLD_FOR_INFORMATION := 'Max value was exceeded. Set hold to 9999 minutes.', only if the duration is greater than MAXINT (2147483647).

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
Property not existing	SysVars	Error. Throws acHoldFor fault.
System variable not existing	SysVars	Error. Throws acHoldFor fault.
Invalid duration (non-numeric)	SysVars	Error. Throws acHoldFor fault.
Duration less than 0	SysVars	Error. Throws acHoldFor fault.
Duration := 0	SysVars	Success. No interruption.
Duration > MAXINT (2147483647)	SysVars	Success. Interruption for 9999 minutes.

Policies: none

1.2.16 acSendToBatchQueue

Caption: Send To Batch Queue.

Throws Fault=acSendToBatchQueue.

The *acSendToBatchQueue* activity is an activity that creates an order containing the current process attachment in the batch queue. Uses the *createOrder* operation from the *BatchQueue* component (refer to chapter *3.1.1 createOrder* of the *BIS6 Base Components* manual for more details).

Field name	Value	Description
Name	*<Text>*[1]	Mandatory. Editable text field. Default value '$[TRACKID]'. The name of the order. The *Name* supports: • System variables (page 392), e.g. '${ORDER_NAME}'. • Properties (page 392), e.g. '$[ORDER_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Stored as $[BATCH_NAME] property.
Group	*<Text>*[1]	Optional. Non-editable text field. The assigned group of the order. The *Group* supports: • Properties (page 392), e.g. '$[ORDER_GROUP]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). System variables are not supported. Stored as $[BATCH_GROUP] property.
Queue	*<Ref>*[2]	Mandatory. Non-editable selection field. Refer to Common Fields (page 1). Stored as $[BATCH_QUEUE] property.
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.
2) Reference to master data record.

Preconditions:

- Available process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- BATCH_NAME := Order name, value from *Name* field.
- BATCH_GROUP := Order group, value from *Group* field.
- BATCH_QUEUE := Order queue, value from *Queue* field.
- BATCH_CORRELATION_ID := Correlation ID of batch order.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid system variable in *Name* field	Error. Throws acSendToBatchQueue fault.
Invalid property in *Name* or *Group* field	Success. Order name is used as entered into field.

Policies: none.

1.2.17 acSetProperties

Caption: Set properties.

Throws Fault=acSetProperties.

The *acSetProperties* activity is an activity that sets up to 5 custom properties (page 392) that can be used later on in the process.

Field name	Value	Description
Key value pair 1	*<KVPair>*[1]	Mandatory. Custom property defining key/property name and value. Invalid characters ", &, < cannot be entered. The *Key value pair* supports: • System variables (page 392), e.g. '${MY_PROP_NAME}=test*.dta'. • Properties (page 392), e.g. '$[MY_PROP_NAME]=test*.dta'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Use self-defined properties only and check the names carefully because

Field name	Value	Description
		existing properties and also system properties are overwritten without any warnings. Recommendation: Use prefixes for property names. Recommendation: Never overwrite system properties.
Key value pair 2	*<KVPair>*[1]	Optional. Custom property. Refer to *Key value pair 1*.
Key value pair 3	*<KVPair>*[1]	Optional. Custom property. Refer to *Key value pair 1*.
Key value pair 4	*<KVPair>*[1]	Optional. Custom property. Refer to *Key value pair 1*.
Key value pair 5	*<KVPair>*[1]	Optional. Custom property. Refer to *Key value pair 1*.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Property consisting of key and value separated by '='. Key and value are arbitrary names except invalid characters: ", &, <.
2) Arbitrary text.

Preconditions:

- None.

Result:

- Current process attachment: not accessed and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 100.
- Property with key and value from each key-value pair.

Possible cases of failure:

Possible case of failure	Success or error behavior
Missing key	Success. No property set.
Key consisting of blanks	Success. No property set.
Missing value	Success. Property set.
Missing '='	Success. No property set.
Key already existing as property name	Success. Existing property is overwritten.

Policies: none

1.2.18 acSetInfo

Caption: Set Inspector info fields.

Throws Fault=acSetInfo.

The *acSetInfo* activity is an activity that sets 2 info fields 'Info 14' and 'Info 15' for process monitoring.

⚠ The process monitoring 'Info<n>' fields are restricted to 128 characters. Longer texts in *Info14* or *Info15* fields that are created by property and/or XPath expression resolving are cut without warning.

Field name	Value	Description
Info14	*<Text>*[1]	Optional. Information text for display in process monitoring. If this field is empty, no value will be set for process monitoring, i.e. a previously set value is not overwritten. A previously set text can be used by inserting $[INFO14] property. The *Info14* supports: • Properties (page 392), e.g. '$[MY_BOOKMARK_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). • Maximum text length is 128. System variables are not supported.
Info15	*<Text>*[1]	Optional. Information text for display in process monitoring. If this field is empty, no value will be set for process monitoring, i.e. a previously set value is not overwritten. A previously set text can be used by inserting $[INFO15] property. The *Info15* supports: • Properties (page 392), e.g. '$[MY_BOOKMARK_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). • Maximum text length is 128. System variables are not supported.
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.

Preconditions: none.

Result:

- Process monitoring info fields 14 and 15 filled.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 100.
- INFO14 := Content of *Info14*.[1]
- INFO15 := Content of *Info15*.[1]

1) The text fields do not allow more than 128 characters. Thus, text length is generally restricted to 128 characters, too. But if the text is originated from resolving properties and/or XPath expressions it can be longer. In this case, it is cut without warning when entering the text into the process monitoring record.

Possible cases of failure:

Possible case of failure	Success or error behavior
Empty text	Success. Info field not set. Any previous text is kept.
Text with 1 or more blanks.	Success. Info field set. Existing text is overwritten.
Final[1] text is longer than 128 characters	Success. Info field set.Text is cut after 128 characters.

1) Text after properties and XPath expressions have been fully resolved.

Policies: none.

1.2.19 acTracker

Caption: Tracker.

Throws Fault=acTracker

The *acTracker* activity is an activity that stores custom information to message tracking (MT). Uses the *execute* operation *of the BicProcessor* component (refer to the chapter *4.2.1 execute* of the *BIS6 Base Components* manual *(link*) or to the *BIC Operation* manual manual for more details*).*

Mode of operation:
All process properties and the activity properties are passed to the selected mapping up to a maximum number of 2000. The mapping accesses these properties using the getInputValue(“KEY_NAME”) method and creates an MT entry. This entry is stored to the database in consideration of the primary keys cWfId and cClient. The process writes its standard MT entry generally using the $[TRACKID] as cWfId and an empty cClient (displayed as '#unknown#' in MT). If the *Line of business* is given in the process, it is stored as $[LINE_OF_BUSINESS]. cClient should be set to $[LINE_OF_BUSINESS] in this case (be aware that cClient can be configured as text or as numbers). If there is no *Line of business* given cWfID must be different from the standard process MT entry: It is recommended to set '$[TRACKID].$[LINE_OF_BUSINESS]' as cWfId in the mapping resp. something like '$[TRACKID].public' for different primary keys and also a clearer arrangement in the MT overview table.
The result properties of the mapping are passed back to the process up to a maximum number of 2000.

Field name	Value	Description
Tracker mapping		Mandatory. Editable selection field. List of BIC mappings. Be careful, the selection list shows all available mappings, not only the MT mappings. The *Tracker mapping* supports: • Properties (page 392), e.g. '$[MY_MAPPING]'. • System variables (page 392), e.g. '${MY_MAPPING}'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). A *Tracker mapping* template is available in <BIS_ROOT>/software/mt/map_MT_MappingTemplate.xml.
Tracking status	*<Status>*[1]	Mandatory. Non editable selection field. Default value INITIAL. Passed to the mapping which can access them using getInputValue("MT_STATUS"). Stored to MT_STATUS property.
Additional parameter	*<KVList>*[2]	Optional. Editable text field. Default value empty. Parameter list. Passed to the mapping which can access them using getInputValue("PARAMS").

Field name	Value	Description
		The *Additional parameter* supports: • Properties (page 392), e.g. '$[PARAM_LIST]'. • System variables (page 392), e.g. '${PARAM_LIST}'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Stored to $[PARAMS] property. ℹ This parameter list is passed to the mapping as a complete list in the $[PARAMS] property. Accessing the parameters inside must be done by the mapping.
Create file link		Optional. Non editable selection field. Provides several modes for creating file links in MT. Usually, MT provides 3 file links that can be used to attach files to the MT information.
	<don't link file>	Default. Do not attach the current process attachment.
	<automatic link>	Attaches the current process attachment using an automatic link. The link name is generated as '$[TRACKID].${COUNTER}'. The target link is fetched from *Link index*. Makes *Link index*, and *Store queue name* visible.
	<custom link>	Attaches the current process attachment using a customizable link, i.e. the link name is entered into *Link name*, and the target link is fetched from *Link index*. Makes *Link index*, *Link name* and *Store queue name* visible.
Link index		Optional. Editable selection field. Default value 'cLink1'. Provides 'cLink1', 'cLink2' and 'cLink3' corresponding the 3 file links provided by MT. Determines the message tracking file link field where the current process attachment is stored. The *Link index* supports: • Properties (page 392), e.g. '$[MY_LINK_INDEX]'. System variables are not supported. Stored to LINK_INDEX property. Visible when *Create file link* is set to '<automatic link>' or '<custom link>'.
Link name		Mandatory. Editable text field. Default value '$[TRACKID].${COUNTER}'. The link name to use when the current process attachment is attached to the MT information as a <custom link>. Visible when *Create file link* is set to <custom link>. The *Link name* supports: • System variables (page 392), e.g. '${MY_LINK_NAME}'. • Properties (page 392), e.g. '$[MY_LINK_NAME]'. In addition, the current process attachment is stored using *Link name* as file name to a sub folder below the path specified by the system variables: • ${SSC-MT-ROOT-PATH}/${MFT-MT-DATA-SUB-PATH}/

Field name	Value	Description
		Stored to LINK_NAME property.
Additional link files	*<KVList>*[2]	Optional. Editable text field. Default value ';' corresponding to an empty list. Key-value list assigning available bookmarks (page 6) to the MT file links, e.g. 'cLink1=AccompanyingPDF;cLink3=DespositPDF;'. The key is one of the available MT file links 'cLink1', 'cLink2', 'cLink3'. The value is an available bookmark. The link file names of these file links cannot be set and are always generated as '$[TRACKID].${COUNTER}'. Take care to not overwrite *Link index* if used. The *Additional link files* support: • Maximum number of bookmarks is 3. Additional bookmarks are ignored. • System variables (page 392), e.g. '${MY_LINKS}'. • Properties (page 392), e.g. '$[MY_LINKS]'.
Link file handling		Optional. Non-editable selection field. Property with three different modes that shows how to handle the case when a link file has already been created. In each case, the link file is created if it does not yet exist.
	Compare file with existing	Default. If a file with the name of the link file already exists, this existing file and the new file are compared: • Identical file content: This activity finishes with success. A different timestamp is ignored. • Different file content: This activity stops with error, throws an *acTr acker* fault.
	Fail on exist	If a file with the name of the link file already exists, the activity stops with error and throws an *acTracker* fault.
	Overwrite file if exists	If a file with the name of the link file already exists, it is overwritten.
Report tracker conversion log as bookmark	True	Creates a bookmark named 'TRACKER_LOG' containing the conversion log.
	False	Conversion log is not bookmarked.
Store queue name	<Ref>[3]	Mandatory. Visible when *Create file link* is set to '<automatic link>' or '<custom link>'. Refer to Common Fields (page 1).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Available status: INITIAL, RUNNING, SUCCESS, PENDING, TERMINATED, STOPPED, ERROR, SUSPECT, CORRUPT.
2) Semicolon separated key-value list.
3) Reference to master data.
4) Arbitrary text.

Additional configuration:

System variable	Description
SSC-MT-ROOT-PATH	Valid MT root path for log and data files.
MFT-MT-DATA-SUB-PATH	Valid sub path for MT data files.

Preconditions:

- StoreService adapter started: if *Create file link* is switched to <automatic link> or <custom link>.
- Available process attachment.

- Optional *Line of business* as first entry of the entity.

Result:

- Current process attachment: unchanged.
- Process status information written into message tracking with the process TRACKID as primary key. It is expected that TRACKID is not used for the acTracker mapping.
- Additional attachments if created by the mapping.

Sets the following properties:

- PARAMS := content of *Additional parameter*.
- STATUS := selected value of *Tracking status*.
- MT_STATUS := selected value of *Tracking status*.
- MAPPING := selected value of *Tracker mapping*.
- TRACKER_MAPPING_NAME := selected value of *Tracker mapping*.
- MT_FOLDER := MT data folder from system variable ${MFT-MT-DATA-SUB-PATH}.
- LINE_OF_BUSINESS := line of business information.
- LINK_INDEX := selected value of *Link index* if *Create file link* is set to '<custom link>' or '<automatic link>'.
- LINK_NAME := either entered *Link name* if *Create file link* is set to '<custom link>' or the automatically generated link name if *Create file link* is set to '<automatic link>'.
- TARGET_NAME := path and name of last link file
- TRACKER_LOG := bookmarked conversion log (available if *Report tracker conversion log as bookmark* is switched on).
- Response properties from the *Tracker mapping*. Maximum number is 2000. Additional properties are ignored.

Possible cases of failure:

Possible case of failure	**Success or error behavior**
Invalid mapping name	Error. Throws acTracker fault.
Conversion error in mapping	Error. Throws acTracker fault.
Primary keys identical to standard MT entry	Success. No data stored to database.
Invalid *Link index*	Success. No data stored to database.
Invalid link index in Additional link files	Success. No data stored to database.
Not existing bookmark in *Additional link files*	Error. Throws acTracker fault.

Policies: none.

1.3 File Content Activities

This section describes file content related activities.

1.3.1 acAdjustLineFeed

Caption: Adjust line feed.

Throws Fault=acAdjustLineFeed.

The *acAdjustLineFeed* activity is a single file activity that reformats a file, namely modifies or removes line feed characters and adapts line length of the current process attachment. Uses the *transform* operation of the *Tran sformer* component (refer to chapter *5.2.1 Transform* of the *BIS6 Base Components* manual for more details).

Field name	Value	Description
Line termination		Mandatory. Non editable selection field. Several modes for handling line breaks configuration: Modifies or removes line feed characters and formats line length.
	Replace LF with CRLF	Default. Replaces each Line Feed character by a Carriage Return Line Feed character sequence. Expects Line Feed characters as line breaks.
	Replace CRLF with LF.	Replaces each Carriage Return Line Feed character sequence by a single Line Feed characters
	Remove CRLF	Removes all Carriage Return Line Feed character sequences.
	Remove LF	Removes all Line Feed characters.
	Fix length record, CRLF	Reformats each line to the length set by *Record length* filling additional space with blanks. Expects Carriage Return Line Feed character sequences as line breaks. Makes *Record length* visible.
	Fix length record, LF	Reformats each line to the length set by *Record length* filling additional space with blanks. Expects Line Feed characters as line breaks. Makes *Record length* visible.
	Fix length record, CRLF using filler	Reformats each line to the length set by *Record length* filling additional space with *Filler*. Expects Carriage Return Line Feed character sequences as line breaks. Makes *Record length* and *Filler* visible.
	Fix length record, LF using filler	Reformats each line to the length set by *Record length* filling additional space with *Filler*. Expects Line Feed characters as line breaks. Makes *Record length* and *Filler* visible.
Filler		Default 'Space'. Editable selection field. Provides predefined entries 'Space' and 'Hash'. Visible when *Line termination* set to a 'Fix length record, ... using filler' entry. • 'Space' fills a short line with spaces ' '. • 'Hash' fills a short line with hash marks '#'. • All other entries are used as they are, for instance: entry 'Has' fills 5 positions like this 'HasHa'. Properties, system variables and xpath expressions are not supported.
Record length	*<Int>*[1]	Mandatory. Editable number field. Default value 80. Visible when *Line termination* set to a 'Fix length record, ...' entry.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

[1] Positive integer.
[2] Arbitrary text.

Preconditions:

- Available process attachment.

Result:

- New process attachment: formatted content.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := used line terminator depending on *Line termination*, one of: REPLACELFWITHCRLF, REPLACECRLFWITHLF, REMOVELF, FIXLENGTHLFFILLER, FIXLENGTHCRLFFILLER, FIXLENGTHLF, FIXLENGTHCRLF.

Policies:

ALLOWANCE/SYSTEM-EXECUTE-SCRIPTS

1.3.2 acAppendBookmarks

Caption: AppendBookmarks.

Throws Fault=acAppendBookmarks.

The *acAppendBookmark*s activity is a single file activity that creates a new process attachment from a list of bookmarks. Booksmarks can be created using the *scBookmark* (page 6) activity.

Field name	Value	Description
Bookmarks to append	<*Bookmark* List>[1]	Mandatory. Default value 'NOT_SET'. List of bookmark names. Invalid characters ", &, < cannot be entered: The list of bookmarks is processed in the given order: • An empty list creates an empty process attachment. • '<current>' uses the current process attachment. • Maximum number of bookmarks is 2000. Additional bookmarks are ignored. Supports: • System variables (page 392), e.g. '${MY_BOOKMARK_NAME}'. • Properties (page 392), e.g. '$[MY_BOOKMARK_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Log message	<*Text*>[2]	Optional. Refer to Common Fields (page 3).

1) Semicolon separated list of bookmark names i.e. arbitrary names except invalid characters ", &, <.
2) Arbitrary text.

Preconditions:

- Available process attachment, only if '<current>' is part of the bookmarks list.

Result:

- New process attachment: contains all listed bookmarks, empty for an empty list.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 100.

Possible cases of failure:

Possible case of failure	Success or error behavior
Empty bookmark list	Success. Creates empty process attachment.
Not existing bookmark in bookmark list	Error. Throws acAppendBookmarks fault.
Bookmark name containing underscore (_), colon (:), hash mark (#), exclamation mark (!), question mark (?), asterisk (*), equal sign (=), greater-than sign (>), dot (.)	Success. Creates process attachment.
Bookmark name containing comma (,), semicolon (;)	Error. Throws acAppendBookmarks fault.

Policies: None.

1.3.3 acAppendToDocument

Caption: AppendToDocument.

Throws Fault=acAppendToDocument.

The *acAppendToDocument* activity is a single file activity that allows inserting a header and appending a trailer to the current attachment. Uses the *pickupFile* operation of the *Pickup Service* component (refer to chapter *6.5.1 pickupFile* of the *BIS6 Base Components* manual for more details).

Field name	Value	Description
Add header		Mandatory. Selection field. Provides 2 header modes.
	No header	Default. Do not add a header.
	Text	Enter the header text directly. Makes the *Header text* visible.
	File	Use the content of a file as header text. Makes the *Header text* visible.
Header text	*<Text>*[1]	Mandatory. The text to be used as header for the current process attachment. Visible when *Add header* is set to 'Text'. The *Header text* supports: • System variables (page 392), e.g. '${MY_HEADER}'. • Properties (page 392), e.g. '$[MY_HEADER]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).

Field name	Value	Description
		• URL encoding, e.g. '%7B' meaning '{' and '%7D' meaning '}'. The text is directly inserted at the beginning of the current process attachment without any whitespace or line break. If a line break is desired HTML code is used to insert it: • Direct HTML code: ‘&#13;&#10;’. Note that the hexadecimal notation ‘&#xD;&#xA;’ does not work without an XPath expression (see below). • XPath expression with HTML code: $(concat(‘This is the header’, ‘&#13;&#10;’)) or $(concat(‘This is the header’, ‘&#xD;&#xA;’)). URL encoding in the header or trailer is allowed.
Header path	*<FilePath>*[2]	Mandatory. The path of the file whose content is to be inserteded as header at the beginnging of the current process attachment. Visible when *Add header* is set to 'File'. The *Header path* supports: • System variables (page 392), e.g. '${MY_HEADER_FILE}'. • Properties (page 392), e.g. '$[MY_HEADER_FILE]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). The *Header path* file content is directly inserted at the beginning of the current process attachment without any whitespace or line break. The *Header path* file content supports: • System variables (page 392), e.g. '${MY_HEADER}'. • Properties (page 392), e.g. '$[MY_HEADER]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). • URL encoding, e.g. '%7B' meaning '{' and '%7D' meaning '}'.
Add trailer		Mandatory. Selection field. Provides 3 trailer modes.
	No trailer	Default. Do not append a trailer.
	Text	Enter the trailer text directly. Makes the *Trailer text* visible.
	File	Use the content of a file as trailer text. Makes the *Trailer text* visible.
Trailer text	*<Text>*[1]	Mandatory. The text to be appended to the current process attachment. Visible when *Add trailer* is set to 'Text'. The *Trailer text* supports: • System variables (page 392), e.g. '${MY_TRAILER}'. • Properties (page 392), e.g. '$[MY_TRAILER]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). • URL encoding, e.g. '%7B' meaning '{' and '%7D' meaning '}'.

Field name	Value	Description
		The text is directly appended to the end of the current process attachment without any whitespace or line break. If a line break is desired HTML code is used to insert it: • Direct HTML code: ‘&#13;&#10;’. Note that the hexadecimal notation ‘&#xD;&#xA;’ does not work outside of an XPath expression (see below). • XPath expression with HTML code: $(concat(‘&#13;&#10;’, ‘This is the header’)) or $(concat(‘&#xD;&#xA;’, ‘This is the header’)).
Trailer path	*<FilePath>*[2]	Mandatory. The path of the file whose content is to be appended as trailer to the current process attachment. Visible when *Add trailer* is set to 'File'. The *Trailer path* supports: • System variables (page 392), e.g. '${MY_TRAILER_FILE}'. • Properties (page 392), e.g. '$[MY_TRAILER_FILE]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). The *Trailer path* file content is directly appended to the end of the current process attachment without any whitespace or line break. The *Trailer path* file content supports: • System variables (page 392), e.g. '${MY_TRAILER}'. • Properties (page 392), e.g. '$[MY_TRAILER]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). • URL encoding, e.g. '%7B' meaning '{' and '%7D' meaning '}'.
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.
2) Valid file path and valid file name.

Preconditions:

- Available process attachment with the encoding set.

Result:

- New process attachment.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 'Executed'.

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
Current attachment without encoding		Error. Throws acAppendToDocument fault.
No header and no trailer set		Success. Process attachment is unchanged.
Directory for header file not existing	Add header - file	Error. Throws acAppendToDocument fault.
Header file not existing	Add header - file	Error. Throws acAppendToDocument fault.
Header file without permissions	Add header - file	Error. Throws acAppendToDocument fault.
Header file locked	Add header - file	Error. Throws acAppendToDocument fault.
Directory for trailer file not existing	Add trailer - file	Error. Throws acAppendToDocument fault.
Trailer file not existing	Add trailer - file	Error. Throws acAppendToDocument fault.
Trailer file without permissions	Add trailer - file	Error. Throws acAppendToDocument fault.
Trailer file locked	Add trailer - file	Error. Throws acAppendToDocument fault.

Policies:

ALLOWANCE/SYSTEM-EXECUTE-SCRIPTS

1.3.4 acCalculateHash

Caption: Calculate hash.

Throws Fault=acCalculateHash.

The *acCalculateHash* activity is a single file activity that determines the hash value of the current process attachment. The determined hash value can be used by the *acDuplicateCheck* (page 57) activity in order to check if the current process attachment is a duplicate.

Field name	Value	Description
Hash type		Mandatory. Non editable selection field. Provides the available hash algorithms.
	IRMark	British Inland Revenue Mark.
	MD5	Default. Message-digest alorithm 5. Produces a 128-bit hash value.
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.

Preconditions:

- ScriptExecutor adapter started.
- Available process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 100.
- HASH_TYPE := The selected hash type.
- HASH_VALUE := The determined hash value.

Policies: none.

1.3.5 acDuplicateCheck

Caption: DuplicateCheck.

Throws Fault=acDuplicateCheck.

The *acDuplicateCheck* activity is an activity that allows checking for a duplicate ID inside the system, for example the MD5 Hash code of a file. Uses the *doDuplicateCheck* operation of the *BIC Duplicate Checker* component (refer to chapter *4.5.1 doDuplicateCheck* of the *BIS6 BaseComponents* manual or to the *BIC Duplicate Checker Module* manual) for more details).

Mode of operation:
The *acDuplicateCheck* activity expects a unique *Duplicate check ID* that uniquely identifies the incoming data. Each check is stored as an entry in the database. All these entries are used for the duplicate checks. A check returns 'DUPLICATE_DETECTED := YES' if a matching entry is found in the database. Reorganization is configured by the scheduler task 'BIC_DUP_CHECKER_FINAL_CLEANER'. Default reorganization time is 15 days. Refer to *BIC Duplicate Checker Reorganization* for more details.

Field name	Value	Description
Duplicate check ID	*<UniqueID>*[1]	Mandatory. Editable text field. Empty by default. A string that is used for duplication checking. Can be a hash of the content of the current process attachment or some unique concatenation of process and process attachment properties. Hash value: Unique representation of the content of the current process attachment, e.g. *MD5* hash of the content of the current attachment. This hash can be calculated using the *acCalculateHash* (page 56) activity; the resulting hash value is stored in the $[HASH_VALUE] property. ID concatenation: Unique representation of the process attachment properties, e.g. $[FROM_PATH] or $[MESSAGE_TYPE] depending on the attachment type and adapter type. The *Duplicate check ID* supports: • System variables (page 392), e.g. '${MY_ID}'. • Properties (page 392), e.g. '$[MY_ID]' or $[HASH_VALUE] if the *acCalculateHash* activity has been used before. • XPath expressions (page 392), e.g. $(seeutil:getContextId()), $(seeutil:getProcessId()), $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). • Minimum length 1, maximum length 400. • Trimming of white spaces, e.g. leading and trailing blanks are removed.

Field name	Value	Description
Duplication check mode		Mandatory. Non-editable selection field. Duplication check modes.
	CHECK	Default. Standard mode for 'normal' checks.
	RESET	Any existing entries with identical *Duplicate check ID* are deleted from database before performing the check. Thus, this check always results in 'No duplicate detected'. If there is no previous entry with an identical *Duplicate check ID* this activity fails.
Handle duplicate		Mandatory. Non editable selection field. Provides 2 modes what to do if a duplicate is found. Irrelevant when *Duplication check mode* is set to 'RESET'.
	Fail on duplicate	Default. Process stops with error if a duplicate is found.
	Report duplicate	Process continues with success. Result of the duplicate check is available as property $[DUPLICATE_DETECTED]. 'YES' if a duplicate has been found. 'NO' if no duplicate has been found
Duplicate check type		Determines base parameter of the *BIC Duplicate Checker* component duplicationHash and duplicationID both of which can be used for duplication checking.
	Duplication hash	Uses duplicationHash parameter of the *BIC Duplicate Checker* component. Note that this parameter is deprecated. It is recommended to use the Duplication ID type instead.
	Duplication ID	Default. Uses duplicationID parameter of the *BIC Duplicate Checker* component.
Allow redecide	True	Default. When a process is redecided this is not marked as duplication.
	False	When a process is redecided this is marked as duplication.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Unique ID representing the process and the current process attachment.
2) Arbitrary text.

Preconditions:

- Current process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- DUPLICATE_CHECK_ID := The ID checked for duplicates.
- DUPLICATE_DETECTED := 'YES' if a duplicate is found.
  'NO' if no duplicate is found.
- STATUS := 'SUCCESS'.
- RESULT := 100.

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
Duplicate ID detected	Fail on duplicate	Error. Throws acDuplicateCheck fault.
Duplicate ID detected	Report duplicate	Success. DUPLICATE_DETECTED := 'YES'.
Duplicate ID detected	RESET	Success. DUPLICATE_DETECTED := 'NO'. Deletes previous entry for duplicate ID.
No Duplicate ID detected	RESET	Error. Throws acDuplicateCheck fault.
Duplicate ID longer than 400 characters		Error. Throws acDuplicateCheck fault.
Duplicate ID with blanks only		Success. No check performed. DUPLICATE_DETECTED := 'NO'.
Non-existing property in *Duplicate check ID*		Success. *Duplicate check ID* value used as is.
Non-existing system variable		Error. Throws acDuplicateCheck fault.

Policies: none.

1.3.6 acClassify

1.3.6.1 acClassify

Caption: Classify.

Throws Fault=acClassify.

The *acClassify* activity is a single file activity that calls the Classify operation of the Classifier component.

Mode of operation:
There is no configuration. Main purpose is to store the result values as properties in the process properties bag.

Field name	Value	Description
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.

Preconditions:

- Available process attachment.

Result:

- Current process attachment, not changed.

Properties:

- STATUS := 'SUCCESS'.
- CLASSIFICATION_FILE_TYPE = detected file type.
- Result properties from the Classify operation (depending on the classification results), for instance: CHARACTER_ENCODING, COMPRESSION_METHOD, COMPRESSED, SEGMENT_DELIMITER

1.3.7 acCodePageConverter

Caption:CodepageConverter.

Throws Fault=acCodePageConverter.

The *acCodePageConverter* activity is a single file activity that converts the current process attachment from a source code page to a target code page. Uses the *convert* operation of the *CPConverter* component (refer to chapter *5.5.1 convert* of the *BIS6 Base Components* manual for more details).

Field name	Value	Description
Source code page		Mandatory. Editable selection field. Code page of current process attachment before conversion. Provides automatic code page detection and explicit code page selection as well as manual input of code pages not covered by the drop-down list.
	<from attachment>	Automatic code page detection of current process attachment: Looks for character set declaration inside. Code page detection does not work for process attachments with binary or missing encoding. It is recommended to use the '<from classifier>' switch in this case.
	<from classifier>	Automatic code page classification of current process attachment: Tries to 'guess' the code page for process attachments where the encoding is set to <binary>.
	<from XML declaration encoding>	Automatic code page detection of current process attachment in case it is an xml file: Looks for code page declaration in its xml root.
	Cp037	Default. IBM code page 037. EBCDIC code page with full Latin-1 character set.
	<CodePage>[1]	Bunch of code pages, some available from drop-down list. Refer to sub-section **Available code pages** below for an enumeration of these available code pages.
Target code page		Mandatory. Editable selection field. Code page of new process attachment after conversion. Provides automatic code page detection and explicit code page selection as well as manual input of code pages not covered by the drop-down list.
	<from classifier>	Automatic code page classification of current process attachment: Tries to 'guess' the code page for process attachments where the encoding is set to <binary>. Actually same as in *Source code page*. If an attachment has no encoding set, using '<from classifier>' in *Source code page* and *Target code page* sets the 'guessed' code page to the attachment.
	<from XML declaration encoding>	Automatic code page detection of current process attachment in case it is an xml file: Looks for code page declaration in its xml root.

Field name	Value	Description
	Cp037	Default. IBM code page 037. EBCDIC code page with full Latin-1 character set.
	<CodePage>[1]	Bunch of code pages, some available from drop-down list.
Abort on error	True/False	Mandatory. Refer to Common Fields (page 3).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Available valid code page. Registered character sets can be looked up at *http://www.iana.org/assignments/character-sets*.
2) Arbitrary text.

Preconditions:

- Available process attachment.

Result:

- New process attachment: with target code page.

The following properties are set:

- STATUS := 'SUCCESS'.
- RESULT := 100.
- SOURCE_CP := Source code page.
- TARGET_CP := Target code page.

Possible cases of failure:

Possible case of failure	Source code page	Success or error behavior
Invalid source code page		Error. Throws acCodePageConverter fault.
Process attachment without code page	'<from attachment>'	Error. Throws acCodePageConverter fault.
Classifier returns invalid code page	'<from classifier>'	Error. Throws acCodePageConverter fault.
Invalid target code page		Error. Throws acCodePageConverter fault.

Policies: None.

Available code pages

- Cp1148 (Cp500 + EURO)
- Cp1047 (EBCDIC Latin-1)
- Cp037
- Cp850
- US_ASCII
- Windows-1252
- ISO-8859-1
- SeeCpEBCDIC (Note: available in Source code page only)

- SeeCp500 (Note: available in Source code page only)
- ISO-8859-15
- UTF-8
- UTF-16
- UTF-16BE
- UTF-16LE
- UnicodeBig
- UnicodeLittle
- windows-1250
- windows-1251
- windows-1253
- windows-1254
- windows-1257
- ISO-8859-2
- ISO-8859-4
- ISO-8859-5
- ISO-8859-7
- ISO-8859-9
- ISO-8859-13
- KOI8-R
- windows-1255
- windows-1256
- windows-1258
- ISO-8859-3
- ISO-8859-6
- ISO-8859-8
- windows-31j
- EUC-JP
- x-EUC-JP-LINUX
- Shift_JIS
- ISO-2022-JP
- x-mswin-936
- GB18030
- x-EUC-CN
- GBK
- ISCII91
- x-windows-949
- EUC-KR
- ISO-2022-KR
- x-windows-950
- x-MS950-HKSCS
- x-EUC-TW
- Big5
- Big5-HKSCS
- TIS-620

1.3.8 acContentAnalyzer

Caption: acContentAnalyzer.

Throws Fault=acDeleteFile.

The *acContentAnalyzer* activity is a single file activity that extracts fields from the current process attachment which are stored as properties (page 392). Uses the *Extract* method from the *BIS File Content Extractor* component *(*refer to the *BIS6 Base Components* manual *(link)* chapter *4.4.1 Extract* ## ## ### ### #### ####### ######### ######).

Field name	Value	Description
Analyzer configuration	*<Ref>*[1]	Mandatory. Editable selection field. Reference of BIC extractor configuration. Configures the fields that are to be extracted from the current process attachment. The extracted fields are stored as properties (page 392) using the configured names. They can be used later on in the process. Properties, system variables and XPath expressions are not supported.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.

Preconditions:

- BIS content extractor node started.
- Available process attachment.

Result:

- Current process attachment: not changed.

The following properties are set:

- STATUS := 'SUCCESS'.
- RESULT := 'Analyzed'.
- One property for each extracted field with the field name configured in BIC extractor configuration.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing Field to extract	Success. Property value set to 'NOT_SET'.

Policies:

ALLOWANCE/PAYLOAD-CONTENT-READ

1.3.9 acConverter

Caption: Convert.

Throws Fault=acConverter.

The *acConverter* activity is a single file activity that converts the current process attachment using the specified mapping. Uses the *convertBAPI* operation from the *BicNode* component (refer to chapter *4.1.2 convertBAPI* of the *BIS6 Base Components manual* for more details).

Field name	**Value**	**Description**
Mapping name	*<Ref>*[1]	Mandatory. Editable selection field. Provides the case-sensitive names of available mappings. If necessary the mapping name is prefixed with the project name. The *Mapping name* supports: • Properties (page 392), e.g. '$[MY_MAPPING]'. • System variables (page 392), e.g. '${MY_MAPPING}'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
SAP BAPI mapping	True	Enables SAP BAPI call within the mapping. Makes *SAP address*, *Retry count* and *Retry interval* visible.
	False	Default. Disables SAP BAPI call within the mapping.
SAP address	*<Ref>*[1]	Mandatory. Non editable selection field. Master data SAP address. Configured in the BIS IDOC connector. Visible when *SAP BAPI mapping* is switched on.
Retry count	*<Int>*[2]	Mandatory. Editable text field. Default value 32. Maximum number of attempts a BAPI connection will be retried in case of connection failure before a mapping error is reported. The *Retry count* supports: • Properties (page 392), e.g. '$[COUNT]'. • System variables (page 392), e.g. '${RETRY_COUNT}'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *SAP BAPI mapping* is switched on.
Retry interval	*<Int>*[2]	Mandatory. Editable text field. Default value 600 seconds. Interval in seconds at which to retry BAPI connections from the mapping if they fail. The Retry interval supports: • Properties (page 392), e.g. '$[INTERVAL]'. • System variables (page 392), e.g. '${RETRY_INTERVAL}'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *SAP BAPI mapping* is switched on.
Additional parameter	*<KVList>*[3]	Optional. Editable text field. List of mapping parameters. A set of properties in the form of *name/content* that serve as BIC input parameters. They may be accessed by the mapping. The *Additional parameter* supports:

Field name	Value	Description
		• Properties (page 392), e.g. '$[KVPAIR1];$[KVPAIR2]'. • System variables (page 392), e.g. '${ALLPARAMS1}'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). This parameter list is passed to the mapping as a complete list in the $[PARAMS] property. Accessing the parameters inside must be done by the mapping.
Additional attachments from bookmarks	*<Bookmarks>*[4]	Optional. Editable text field. List of bookmarked attachments with case-sensitive bookmark names. Additional input data attachments for the mapping. The *Additional attachments fom bookmarks* supports: • Properties (page 392), e.g. '$[BOOKMARK1];$[BOOKMARK2]'. • System variables (page 392), e.g. '${BOOKMARK1}'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). • Maximum number of bookmarks is 2000. Additional bookmarks are ignored.
Report additional attachments as bookmark	True	Get additional mapping attachments as bookmarks using the attachment name as bookmark name.
	False	Default. Do not get additional mapping attachments as bookmarks.
Report conversion log as bookmark	True	Get converter log as bookmark with name CONVERSION_LOG.
	False	Default. Do not get converter log as bookmark.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[5]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Positive Integer.
3) Semicolon separated key-value list.
4) Semicolon separated list of case-sensitive bookmark names i.e. arbitrary names except invalid characters ", &, < (which cannot be inserted).
5) Arbitrary text.

Preconditions:

- BIC Client adapter started.
- Available process attachment.
- Valid bookmarked attachments.

Result:

- New process attachment: converted file.
- Conversion log attachment.

Sets the following properties:

- MAPPING_NAME := name of the mapping.
- QUEUE_NAME := name of the queue.
- PARAMS := value of *Additional parameters*, 'NOT_SET' if it is empty.
- STATUS := 'SUCCESS'.
- RESULT := 'Converted'.
- CONVERSION_LOG := bookmark name for converter log, if *Report conversion log as bookmark* is switched on,
  'NOT_SET' if there is no conversion log.
- CONVERSION_TYPE := 'ATTACHMENT' if the mapping creates an output attachment or
  'MULTIPLE_ATTACHMENT' if the mapping creates several output attachments or
  'NO_MESSAGE' if the mapping does not create an output attachment at all.

Error case properties:

- STATUS := 'ERROR'.
- RESULT := '204'.
- CONVERSION_STATUS := 'ERROR'.
- FAILED_ACTIVITY_LOG := Bookmarked log attachment from BIC in case of mapping error.

Possible cases of failure:

Possible case of failure	Success or error behavior
Empty attachment	Success.
Invalid mapping name, mapping not found	Error. Throws acConverter fault.
Invalid bookmark name in *Additional attachments from bookmarks*	Error. Throws acConverter fault.
Mapping error	Error. Throws acConverter fault.

Policies:

ALLOWANCE/PAYLOAD-CONTENT-READ

1.3.10 acJSON-XMLConverter

Caption: JSON-XMLConverter.

Throws Fault=acJSON-XMLConverter.

The *acJSON-XMLConverter* activity is a single file activity that converts XML format into JSON format and vice versa. Uses the *JSON2XML* and *XML2JSON* operations of the *BIS JSON to XML Converter* component (refer to chapter *3 Operation JSON2XML (link)* and chapter *4 Operation XML2JSON (link)* of the *BIC JSON <> XML* manual for more details).

Field name	Value	Description
Direction of conversion		Optional. Non-editable selection field. Provides supported conversion directions.
	JSON2XML	Default. Conversion direction JSON to XML.
	XML2JSON	Conversion direction XML to JSON. Makes *Text element name* visible.

Field name	Value	Description
Conversion mode		Optional. Non-editable selection field. Provides supported conversion modes which define the actual conversion rules to be applied.
	JACKSON	Default. Currently, the only supported conversion mode. This conversion mode may involve loss of data. Refer to chapter *5 JACKSON mode* of the *BIC JSON <> XML* manual for further details.
Output format well formed	True	Default. The output is formatted in order to improve readability. Formatting may vary depending on the selected *Conversion mode*. In case of XML output, each XML element is located at a new line and is indented.
	False	No additional formatting for improving readability.
Text element name		Optional. Editable text field. Empty by default. The name of the JSON key that is used for XML text element values that have attributes. Component internal default value is 'CONTENT'. System variables and properties are not supported. Visible if *Direction of conversion* is set to 'XML2JSON'.
Queue	*<Ref>*[1]	Mandatory. Non-editable text field. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Arbitrary text.

Preconditions:

- BIS JSON Client adapter started.
- Available process attachment: XML or JSON format depending on *Direction of conversion*.

Result:

- New process attachment: specified output format.

Sets the following properties: None.

Possible cases of failure:

Possible case of failure	Direction	Success or error behavior
No XML format	XML > JSON	Error. Throws acJSON-XMLConverter fault.
No JSON format	JSON > XML	Error. Throws acJSON-XMLConverter fault.

Policies: none.

1.3.11 acFindAndReplace

Caption: FindAndReplace.

Throws Fault=acFindAndReplace.

The *acFindAndReplace* activity is a single file activity that performs multiple context-sensitive search and replace operations on the content of the current process attachment.

Mode of operation:
The list of search-replace patterns is handled successively one after the other. When handling one pattern, an occurrence of the search pattern is looked for. If one has been found it is replaced by the replace pattern. The search for the next occurrence continues at the next file position directly after the replacement string.
It is strongly recommended to use process attachment with a character set. If there is no encoding set or if the encoding is '<binary>' the characterset is assumed to be 'System.getProperty("file.encoding")'. If this encoding does not fit this generally results in a spoiled process attachment.

Field name	Value	Description
Find and replace list	*<KVList>*[1]	Optional. Default value 'myfind=myreplace;'. Editable text field. Semicolon separated list of search-replace patterns handled successively. The *Find and replace list* supports: • Properties (page 392), e.g. '$[REPL_PATTERN_1]; $[REPL_PATTERN_2]'. • Empty replacement strings: delete the search string. • Semicolons and other special characters can be referenced by using HTML notation, for instance '#59' in case of a semicolon. This HTML notation can be used as search and as replace pattern. It cannot be combined with other characters: neither patterns such as 'test##59' nor '#59#44' are supported. • Maximum number of key-value pairs is 2000. Additional key-value pairs are ignored. System variables and empty search strings are not supported.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Semicolon separated key-value list. Each key-value pair has a search and a replace pattern.
2) Arbitrary text.

Preconditions:

- Available process attachment: text based file.

Result:

- New process attachment: file with replacements.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 100.
- FIND_AND_REPLACE_LIST := *Find and replace list* value.

Possible cases of failure:

Possible case of failure	Success or error behavior
Syntax errors in the key-value list	Error. Throws acFindAndReplace fault.
Empty key-value list	Success. File is not changed.
Blanks only in key-value list	Success. File is not changed.

Policies: None

1.3.12 acSSCBusinessCheck

Caption: Business validation.

Throws Fault=acBusinessCheck.

Still under construction.

Requires B2B Standard Solution.

The *acSSCBusinessCheck* activity is an activity that checks the current process attachment (EDIFACT message) for compliance. Uses the *CheckCompliance* operation of the *Compliance Check Adapter for Utilities* (refer to the *Compliance Check Adapter* manual for more details).

Field name	Value	Description
Compliance set		Mandatory. Editable text field. The name of the compliance set that contains the message schemas and overlays necessary for the EDIFACT compliance check. The *Compliance set* supports: • Properties (page 392) • System variables (page 392)
Compliance schema		Mandatory. Editable text field. The name of the message schema that is used for the EDIFACT compliance check. The *Compliance schema* supports: • Properties (page 392) • System variables (page 392)
Processing mode		Non editable selection field.
	Reject if errors	Default.
	No partial processing	Makes *Split key for partial processing* and *Address of message ID* visible.
	Accept with errors	
Split key for partial processing		Mandatory. Editable text field. Visible when *Processing mode* is switched to 'Partial processing'. The *Split key for partial processing* supports: • Properties (page 392) • System variables (page 392)
Address of message ID		Mandatory. Editable text field. Visible when *Processing mode* is switched to 'Partial processing'. The *Address of message ID* supports: • Properties (page 392)

Field name	Value	Description
		- System variables (page 392)
Reply mode		Non editable selection field.
	No reply	
	Always reply	Default.
	Reply on errors	
Create transaction details		Optional. Non-editable selection field.
	Default	Default.
	Do not create	
	Create always	
	Create for errors	
Create segment/ element details		Optional. Non-editable selection field.
	Default.	Default.
	Do not create.	
	Create for errors.	
Response message type		Mandatory. Editable selection field. Provides available compliance check result types. Properties and system variables are not supported.
	Functional acknowledgement (CONTRL/997)	X12_997 - Creates a Functional Acknowledgement (997) output to report the results of the syntactical analysis of the inbound message document.
	Application advice (APERAK/824)	Default. X12_824 - Creates an Application Advice (824) output to report the acceptance, rejection, or partial acceptance or rejection of any transaction set of the inbound message document.
	Functional acknowledgement (CONTRL)	CONTRL - Create a CONTRL acknowledgement output to report the results of the syntactical analysis of the inbound message document.
	Application advice (APERAK)	APERAK - Creates an EDIFACT APERAK in case of a compliance error.
	SEEAPPACK	SEEAPPACK - Creates a SEEBURGER acknowledgement output message.
Response message interchange counter name		Optional. Editable text field. Name of counter supplying unique interchange references. Properties and system variables are not supported.
Response message message number		Optional. Editable text field. Message reference for the response message. Properties and system variables are not supported.
Reply process message type		Mandatory. Editable selection field. sReplyProcessType, needed for new MFTEvent with 'CCEVT' adapter, message type set to *fromNam e* field. Properties and system variables are not supported.

Field name	Value	Description
Output report		Mandatory. Non editable selection field. Configures creation of human readable reports in case of any EDIFACT compliance check errors.
	HTML report	Default. Creates an HTML report.
	Custom report	No report is created in case of compliance errors.
Skip report storage	True	Store the report if available.
	False	Default. Do not store the report.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.

Necessary system variables:

- SSC-MT-MAPPING
- SSC-MAX-SYNC-PAYLOAD-SIZE
- SSC-DESTINATION-COMPLIANCECHECKNG-IN-SMALL
- SSC-DESTINATION-COMPLIANCECHECKNG-IN-BIG
- SSC-CC-USE-STRICT-XSD-VALIDATION
- SSC-STORE-TEMPPATH-EXTENSION

Necessary properties:

- MESSAGETYPE
- PartnerEntity/id
- ProcessMTRelativePath
- ORIGINALFILENAME
- ProcessMTPath

Preconditions:

- Requires B2B Standard Solution
- Edifact Compliance Module adapter started. Refer to Compliance Check Adapter Utility.
- BIC Client adapter started.
- BIC Splitter Node adapter started: in case of partial processing.
- StoreService adapter started.
- Current process attachment.

Result:

- New process attachment: aCorrectedPayload.

Sets the following properties:

- MY_KEY := 'acPGPEncryptMD'.

Policies: none.

1.4 Program and Script Calls

This section describes activities that call external programs and internal bean shell scripts.

1.4.1 acRunExternalProgram

Caption: RunExternalProgram.

Throws Fault=acRunExternalProgram.

The *acRunExternalProgram* activity is a single file activity that allows executing a native program. Uses the *executeScript* operation of the *ScriptExecutor* component using a Bean Shell / BSH script to call the external program (refer to chapter *2.2.2 executeScript* of the *BIS6 Base Components* manual for more details).

Mode of operation:
Call of an external program that can use the current process attachment as source file. A target file created by this external program is picked up as new process attachment. If there is no such target file, the source file is returned instead. If both files don't exist, this activity stops with error.
The name of the BSH script that calls the external program is stored in the system variables (page 392) ${MFT-EUN-EXTERNAL-PROGRAM-SCRIPT} or ${MFT-RUN-EXTERNAL-PROGRAM-SCRIPT}. If the first one does not exist, the second one is used.

<table>
<tr><th>Field name</th><th>Value</th><th>Description</th></tr>
<tr><td>Program path</td><td>*<FilePath>*[1]</td><td>Mandatory. Default value '- path to executable -'. Editable text field.
File path to the external program to be called.
The *Program path* supports:
- System variables (page 392), e.g. ${MY_PROG_PATH}.
- Properties (page 392), e.g. $[MY_PROG_PATH].</td></tr>
<tr><td>Program parameter</td><td><CommandLine></td><td>Mandatory. Default value '%SOURCE_FILE% %TARGET_FILE% %PASSWORD%'. Editable text field. Additional parameters to be passed to the external program as its command line.
The *Program parameter* supports:
- System variables (page 392), e.g. ${MY_CMDLINE}.
- Properties (page 392), e.g. $[MY_CMDLINE].
- Special key words that are resolved before program execution (refer to **Special Program Parameter Keywords** section below for more details):
<table>
<tr><th>Key work</th><th>Description</th></tr>
<tr><td>%SOURCE_FILE%</td><td>Placeholder for file path and name of the source file for the external program where the current process attachment is stored temporarily.</td></tr>
<tr><td>%TARGET_FILE%</td><td>Placeholder for file path and name of the target file if the external program creates an output file that is picked up after execution as new process attachment.</td></tr>
<tr><td>%PASSWORD%</td><td>Placeholder for a hidden password. Replaced by *Password*.</td></tr>
</table>
</td></tr>
</table>

Field name	Value	Description
		Example for 7z.exe program: a -y -o C:\ZipRes %TARGET_FILE% %SOURCE_FILE%.
Password	<Password>	Optional. Editable text field. Password for the external program. Only used if *Program parameter* contains the '%PASSWORD%' key. Properties and system variables are not supported.
Source file name	*<FileName>*[2]	Optional. Editable text field. Empty by default. Used for storing current process attachment serving as source file for the external program. Only used if *Program parameter* contains the '%SOURCE_FILE%' key word in order to replace it. The *Source file name* supports: • System variables (page 392), e.g. ${MY_SOURCE_FILENAME}. • Properties (page 392), e.g. $[MY_SOURCE_FILENAME]. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Working directory	*<FilePath>*[3]	Optional. Default value './'. Editable text field. The current attachment (source file) is stored into this working directory or a sub-directory. The *Working directory* supports: • System variables (page 392), e.g. ${MY_WORKDIR}. • Properties (page 392), e.g. $[MY_WORKDIR]. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Fault pattern (regexp)	*<RegEx>*[4]	Optional. Editable text field. Error pattern used to check the result message of the external program in case the exit value equals 0. If it matches the process stops with error. Properties and system variables are not supported.
progTimeout	*<Int>*[4]	Mandatory. Default value 60. Timeout for program execution in seconds.
Debug level		Mandatory. Selection field. Provides 3 debug levels.
	No debug	Standard level. No logging of script call. No console output and no logging attachment.
	Debug	Simple debug level. Script call is logged. No console output. Logging attachment is created. Log saved to disc to main mft.log file as configured by system variable (page 392) ${MFT-LOG-SUB-PATH}.
	Verbose	Elaborate debug level. Script call is logged. Console output is written. Logging attachment is created. Log saved to disc to main mft.log file as configured by system variable (page 392) ${MFT-LOG-SUB-PATH}.
Queue	*<Ref>*[4]	Mandatory. Refer to Common Fields (page 1).
logmessage	*<Text>*[6]	Optional. Refer to Common Fields (page 3).

1) Valid file path and file name.
2) Valid file name.
3) Valid absolute or relative file path.
4) Regular expression.

5) Positive Integer number.
6) Reference to master data.
6) Arbitrary text.

Special Program Parameter Keywords

<table>
<tr><th>Key work</th><th>Description</th></tr>
<tr><td>%SOURCE_FILE%</td><td>Placeholder for file path and name of the source file for the external program where the current process attachment is stored temporarily. Consists of <i>Working Directory</i>, TRACK_ID and <i>Source file name</i>. If <i>Working Directory</i> or <i>Source file name</i> or both contain one or more blanks, this keyword must be surrounded with double quotes "". If <i>Source file name</i> is

• empty:
The file is stored in the <i>Working directory</i> directly. A generated file name consisting of the TRACK_ID and a file extension is used as file name.
• filled:
The file is stored into a sub folder of this <i>Working directory</i> with the current TRACKID as its name. The <i>source file name</i> is used as file name.

I If this placeholder is not used, the source file is not created.

All created files and directories are temporary and are deleted after executing the external program.</td></tr>
<tr><td>%TARGET_FILE%</td><td>Placeholder for file path and name of the target file if the external program creates an output file. Consists of <i>Working Directory</i>, TRACK_ID and a file name, either generated or taken from the source file. If <i>Working Directory</i> or <i>Source file name</i> (if used) or both contain one or more blanks, this keyword must be surrounded with double quotes "". After external program execution has finished, this file is picked up as new process attachment.
<table>
<tr><th>Keyword</th><th>Target file</th><th>Behavior</th></tr>
<tr><td>used</td><td>not created</td><td>If a source file exists, it is used as target file. Otherwise, this activity fails.</td></tr>
<tr><td>used</td><td>created</td><td>An output file with a name generated from the current TRACK_ID is expected in the <i>Working directory</i>. This file is picked up as new process attachment.</td></tr>
<tr><td>not used</td><td>not created</td><td>If a source file exists, it is used as target file. Otherwise, this activity fails.</td></tr>
<tr><td>not used</td><td>created</td><td>this activity expects an output file with the name (and location of) '%SOURCE_FILE%' and picks it up as new process attachment.</td></tr>
</table>
All created files and directories are temporary and are deleted after executing the external program.</td></tr>
<tr><td>%PASSWORD%</td><td>Placeholder for a hidden password. Replaced by <i>Password</i>.</td></tr>
</table>

Preconditions:

- ScriptExecutor adapter started.

- Current process attachment.

Result:

- New process attachment: either external program target file or original source file.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 100.
- TARGET_EXISTS := TRUE.

Possible cases of failure:

Possible case of failure	**Success or error behavior**
Invalid program path or name	Error. Throws acRunExternalProgram fault.
Program path without permissions	Error. Throws acRunExternalProgram fault.
Program without permissions	Error. Throws acRunExternalProgram fault.
Invalid command line	Error. Throws acRunExternalProgram fault.
Source file name consisting of blanks only	Error. Throws acRunExternalProgram fault.
Blanks in Working directory and/or Source file name and no double quotes around %SOURCE_FILE% keyword	Error. Throws acRunExternalProgram fault.
Blanks in Working directory and/or Source file name and no double quotes around %TARGET_FILE% keyword	Error. Throws acRunExternalProgram fault.
Invalid password (if needed)	Error. Throws acRunExternalProgram fault.
No output file by external program	Error. Throws acRunExternalProgram fault.
Output file by external program, %TARGET_FILE% not used	Success. Target file is named %SOURCE_FILE%.
Working directory not existing	Error. Throws acRunExternalProgram fault.
Working directory without permissions	Error. Throws acRunExternalProgram fault.
Invalid regular expression for *Fault pattern*	Error. Throws acRunExternalProgram fault.

Policies: none.

1.4.2 acScript

Caption: Call script.

Throws Fault=acScriptCall.

The *acScript* activity is an activity that executes a bean shell / BSH script from the ressource repository. Uses the *executeScript* operation of the *ScriptExecutor* component (refer to chapter *2.2.2 executeScript* of the *BIS6 Base Components* manual for more details). Refer to *Creating a BSH Script* (page 410) for more details about script implementation.

Mode of operation:

Call of a ressource repository BSH script with up to 5 input parameters and the current process attachment as input attachment (source file). Can return an output attachment (target file), a bunch of output parameters, and a logging attachment.

Field name	Value	Description
Script name	*<FileName>*[1]	Mandatory. Empty by default. Editable text field. The BSH script filename. The script has to be available in the ressource repository in '<LS>/scripts/bsh/1' (The ressource repository search uses the current logical system, the BEANSHELL type, and the name '1/<*Script name*>', where '1' is the fixed and unchangeable version number). The *Script name* supports: • System variables (page 392), e.g. ${MY_BSH_SCRIPT}. • Properties (page 392), e.g. $[MY_BSH_SCRIPT]. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Parameter 1	<Value>	Optional. Empty by default. Editable text field. Script input parameter value. Set as property and script input parameter with key 'PARAM1'. The *Parameter 1* supports: • System variables (page 392), e.g. ${MY_PARAM}. • Properties (page 392), e.g. $[MY_PARAM]. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Parameter 2	<Value>	Optional. Default value 'NOT_SET'. Editable text field. Script input parameter value. Set as property and script input parameter with key 'PARAM2'. The *Parameter 2* supports: • System variables (page 392), e.g. ${MY_PARAM}. • Properties (page 392), e.g. $[MY_PARAM]. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Parameter 3	<Value>	Optional. Script input parameter value. Refer to *Parameter 2*.
Parameter 4	<Value>	Optional. Script input parameter value. Refer to *Parameter 2*.
Parameter 5	<Value>	Optional. Script input parameter value. Refer to *Parameter 2*.
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid script file name with '.bsh' extension.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- Script Executor adapter started.
- Available process attachment.

Result:

- New process attachment: Output file (log file) from script if existing.
  Current process attachment: Not changed when no output file from script.
- Logging attachment if existing.

Sets the following properties:

- SCRIPT_NAME := BSH script name.
- PARAM1 := Value of first script input parameter.
- PARAM2 := Value of second script input parameter.
- PARAM3 := Value of third script input parameter.
- PARAM4 := Value of forth script input parameter.
- PARAM5 := Value of fifth script input parameter.
- Output parameters set by the BSH script (if existing).
- Output attachments set by the BSH script (if existing)

Possible cases of failure:

Possible case of failure	**Success or error behavior**
Non existing script	Error. Throws acScriptCall fault.
Script failure	Error. Throws acScriptCall fault.

Policies:

ALLOWANCE/SYSTEM-EXECUTE-SCRIPT

1.5 CSV Excel Activities

This chapter contains the activities that convert CSV files to Excel files and vice versa.

1.5.1 acCSVToExcel

Caption: CSVToExcel.

Throws Fault=acCSVToExcel.

The *acCSVToExcel* activity is a single file activity that converts a CSV file to excel format. Uses the *executeScript* operation of the *ScriptExecutor* component (refer to chapter *2.2.2 executeScript* of the *BIS6 Base Components* manual for more details).

Field name	**Value**	**Description**
Excel type		Mandatory. Non-editable selection field. Provides excel target file formats.
	Office 2007+ XML	Default. Office 207 format (.xlsx).
	OLE2 Office Documents	OLE2 format (.xls).
Column termination		Mandatory. Non-editable selection field. Provides column termination modes used in the CSV source file.
	Comma	Default. A comma ',' denotes the end of a column.

Field name	Value	Description
	Semicolon	A semicolon ';' denotes the end of a column.
	Tab	A tab denotes the end of a column.
Cell type handling		Mandatory. Non-editable selection field. Provides 2 modes how to handle an excel cell value.
	Always string	Default. Use strings only.
	Number on numeric value and string on other	Use numbers for numeric values. Use strings for all other values.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.

Preconditions:

- Script Executor adapter started.
- mftCSVToExcel.bsh script stored in resrepo.
- Available process attachment: CSV format.

Result:

- New process attachment: resulting excel file.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 100.
- LAST_ACTIVITY := 'Activity[<ActivityStepNo>]'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Other file format than CSV	Success. Result file is unusable.

Policies: none.

1.5.2 acExcelToCSV

Caption: ExcelToCSV.

Throws Fault=acExcelToCSV.

The *acExcelToCSV* activity is a single file activity that allows converting an excel file to csv format. Uses the *executeScript* operation of the *ScriptExecutor* component (refer to chapter *2.2.2 executeScript* of the *BIS6 Base Components* manual for more details).

The conversion differenciates between boolean, numeric and string type cells and ignores cell formulas and cell formatting. The actual cell value is retrieved and not changed in any way, e.g. by rounding.

General cells are interpreted as text cells for components versions 2.4 and higher. This means Excel value '100' results in CSV value '100'. In previous components versions, Excel value '100' results in CSV value '100.0'.

Field name	Value	Description
Excel type		Mandatory. Non editable selection field. Provides available formats for the excel source file.
	Office 2007+ XML	Default. Office 207 format (.xlsx).
	OLE2 Office Documents	OLE2 format (.xls).
Column termination		Mandatory. Non editable selection field. Provides 3 modes of column termination for the CSV target file.
	Comma	Default. A comma ',' denotes the end of a column.
	Semicolon	A semicolon ';' denotes the end of a column.
	Tab	A tab denotes the end of a column.
Line termination		Mandatory. Non editable selection field. Provides 2 different types of line breaks for the CSV target file.
	Carriage return line feed	Carriage return line feed.
	Line feed	Line feed only.
Target date format		Optional. Editable selection field. Provides the date format for the CSV target file. Applied to all numeric Excel cells representing a date and/or time. If *Target date format* is empty the default date format 'M.d.yy' is used. Supports: • Properties (page 392), e.g. '$[DATE_FORMAT]'. System variables and XPaths are not supported.
	M.d.yy	Default. Example: 1.3.15
	EEE d MMM yyyy HH:mm:ss Z	Example: Sa 3 Jan 2015 12:00:00 +0100
	yyyy.MM.dd G 'at' HH:mm:ss z	Example: 2015.01.03 n. Chr. at 12:00:00 MEZ
	YYYY-'W'ww-u	Example: 2015-W01-6
	yyyy-MM-dd'T' HH:mm:ss.SSSXXX	Example: 2015-01-03T12:00:00.000+01:00
	yyyy-MM-dd'T' HH:mm:ss.SSSZ	Example: 2015-01-03T12:00:00.000+0100
	yyyy-MM-dd HH:mm:ss	Example: 2015-01-03 12:00:00

Field name	Value	Description
	yyyyy.MMMMM.dd GGG hh:mm aaa	Example: 02015.Januar.03 n. Chr. 12:00 PM
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.

Preconditions:

- Script Executor adapter started.
- mftExcelToCSV.bsh script stored in resrepo.
- Available process attachment. Excel format consistent with the *Excel type* field.

Result:

- New process attachment: the resulting CSV file.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 100.
- LAST_ACTIVITY := 'Activity[<ActivityStepNo>].

Possible cases of failure:

Possible case of failure	Success or error behavior
Script executor not started	Error. Throws acExcelToCSV fault.
mftExcelToCSV.bsh script not found	Error. Throws acExcelToCSV fault.
Invalid format of current process attachment	Error. Throws acExcelToCSV fault.
Format of current process attachment not consistent with the *Excel type* field	Error. Throws acExcelToCSV fault.
Invalid date pattern in *Target date format*	Error. Throws acExcelToCSV fault.

Policies: none.

1.6 PDF and Jasper Report Activities

This section describes PDF and Jasper report related activities.

1.6.1 acPDFGenerator

Caption: Generate PDF.

Throws Fault=acPDFGenerator.

The *acPDFGenerator* activity is a single file activity that creates a PDF file from the current process attachment. Uses the *GenerateReport* operation of the *Jasper Reports Processor* component (refer to chapter *4.6.1 GenerateReport* of the *BIS6 Base Components* manual or the *BIC Japser Reports Adapter* manual for more details).

Field name	Value	Description
Report name	*<Name>*[1]	Mandatory. Editable selection field. Default value empty. Path and name of the main Jasper report to use for report generation. Provides all successfully deployed Jasper reports. The *Report name* supports: • System variables (page 392), e.g. '${REPORT_NAME}'. • Properties (page 392), e.g. '$[REPORT_NAME]'.
Locale	<Locale>	Mandatory. Editable selection field providing ENGLISH and GERMAN locales. Can be used in the report. The *Locale* supports: • System variables (page 392), e.g. '${LOCALE}'. • Properties (page 392), e.g. '$[LOCALE]'.
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Name of existing Jasper report.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- BIC Jasper Report Processor adapter started.
- Available process attachment. XML format expected.

Result:

- New process attachment: the generated PDF file.

Sets the following properties:

- REPORT_NAME := Entered *Report name*.
- PDFMetaData with sub properties
  CreationDate := Date of document creation,
  PageCount := Number of PDF pages,
  Producer := Name of the report template used,
  ModDate := Date of last document modification (identical to CreationDate),
  Creator := Name of the program/library that generates the PDF.

Possible cases of failure:

Possible case of failure	Success or error behavior
BIS Jasper Reports Processor adapter not started	Error. Throws acPDFGenerator fault.
Empty report name	Error. Throws acPDFGenerator fault.
Invalid report name	Error. Throws acPDFGenerator fault.

Policies: none.

1.6.2 acPDFSignature

Caption: PDFSignature.

Throws Fault=acPDFSignature.

The *acPDFSignature* activity is a single file activity that sign and verifies a PDF document stored as current process attachment. Uses the *signPDFDocument* operation and the *verify PDFDocument* operation of the *PDF Signature* component/*PDF Secure* adapter (refer to chapter *7.1 signPDFDocument (link)* and chapter *7.2 verifyPDFDocument (link)* of the *PDF Signature Module* manual for more details).

Field name	Value	Description
Method		Mandatory. Selection field. Distinguishes between signing a PDF document and verifying a signed document.
	Sign	Default. Signs the PDF document. Uses the *signPDFDocument* operation. Makes *Digest algorith* and *Signature visibility in pdf* visible.
	Verify	Verifies the signature of a PDF document. Expects a signed PDF document. Uses the *verify PDFDocument* operation

Method: Sign

This Sign method signs a PDF document attached as current process attachment. Uses the *signPDFDocument* operation of the *PDF Signature* component/*PDF Secure* adapter (refer to chapter *7.1 signPDFDocument (link)* of the *PDF Signature Module* manual for more details).

Field name	Value	Description
Key alias	*<Ref>*[1]	Optional. Non editable text field with key-alias selection dialog. The alias of the private key/certificate from the Keystore Manager. The *Key alias* supports: • System variables (page 392), ${ALIAS_NAME}'. • Properties (page 392), e.g. '$[ALIAS_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). I If *PDF Signature own address* is given the *Key alias* is ignored.
Digest algorithm		Mandatory. Non editable selection field. Provides a list of supported digest or hash algorithms.
	MD5	Default. Message-digest alorithm 5. Produces a 128-bit hash value.
	RIPEMD160	Race Integrity Primitives Evaluation Message Digest. Produces a 160-bit hash value.
	SHA1	Secure hash algorithm. SHA-1 hash function. Produces a 160-bit hash value.
	SHA128	Secure hash algorithm. SHA-2 hash function. Produces a 128-bit hash value.
	SHA256	Secure hash algorithm. SHA-2 hash function. Produces a 256-bit hash value.

Field name	Value	Description
	SHA384	Secure hash algorithm. SHA-2 hash function. Produces a 384-bit hash value.
	SHA512	Secure hash algorithm. SHA-2 hash function. Produces a 512-bit hash value.
PDF Signature own address	*<Ref>*[1]	Optional.Editable selection field. PDFSecure address master data. Provides bidirectional and own addresses. Overwrites *Key alias*. The *PDF Signature own address* supports: • System variables (page 392), ${OWN_ADDRESS}'. • Properties (page 392), e.g. '$[OWN_ADDRESS]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Document password	*<Password>*[2]	Optional. Editable password field. The document owner password. Properties, system variables and XPath expressions are not supported.
Signature visibility in pdf	True	The signature is integrated as signature field into the PDF document. Makes visible the *Background image of signature*, *Page of signature*, *Horizontal upper right position*, and *Vertical upper right position* fields below.
	False	The signature is not integrated into the PDF document.
Background image of signature	*<FilePath>*[3]	Optional. Editable text field. File path and name of background image for the graphic part of the signature. The *Background image of signature* supports: • System variables (page 392), ${IMAGE_PATH}'. • Properties (page 392), e.g. '$[IMAGE_PATH]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Signature visibility in pdf* is switched on.
Page of signature	*<Int>*[4]	Mandatory. Editable number field. Default value 1. The page where the signature field is placed. Visible when *Signature visibility in pdf* is switched on. Properties, system variables and XPath expressions are not supported.
Horizontal upper right position	*<Int>*[4]	Mandatory. Editable number field Default value 300. Minimum value 1, maximum value 1600. The upper right X coordinate of the signature field. Visible when *Signature visibility in pdf* is switched on. Properties, system variables and XPath expressions are not supported.
Vertical upper right position	*<Int>*[4]	Mandatory. Editable number field Default value 200. Minimum value 1, maximum value 1000. The upper right Y coordinate of the signature field. Visible when *Signature visibility in pdf* is switched on.

Field name	Value	Description
		Properties, system variables and XPath expressions are not supported.
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Logmessage	*<Text>*[5]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Secure password.
3) Valid file path and name.
4) Integer number.
5) Arbitrary text.

Preconditions:

- PDFSecure Module adapter started.
- Available process attachment: PDF document expected.

Result:

- New process attachment: the signed PDF document.

Sets the following properties:

- PDF_VERIFY_RESULT := 'Document verified'.

Possible cases of failure:

Possible case of failure	Success or error behavior
PDFSecure Module adapter not started	Error. Throws acPDFSignature fault.
No key alias and no own address with key alias	Error. Throws acPDFSignature fault.
No private key as key alias	Error. Throws acPDFSignature fault.
Non existing *Background file of signature*	Error. Throws acPDFSignature fault.
Invalid page number	Error. Throws acPDFSignature fault.

Policy: none.

Method: Verify

This Verify method signs a signed PDF document attached as current process attachment. Uses the *verify PDFDocument* operation of the *PDF Signature* component/*PDF Secure* adapter (refer to chapter *7.2 verifyPDFDocument (link)* of the *PDF Signature Module* manual for more details).

PDF signature verification only supports the 'SHA1' digest algorithm. In case of other digest algorithms, this activity method fails.

Field name	Value	Description
Key alias	*<Ref>*[1]	Optional. Editable text field. The alias of the private key/certificate from the Keystore Manager. The *Key alias* supports: • System variables (page 392), ${ALIAS_NAME}'.

Field name	Value	Description
		• Properties (page 392), e.g. '$[ALIAS_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). If *PDF Signature partner address* is given the *Key alias* is ignored.
PDF Signature partner address	*<Ref>*[1]	Optional.Editable selection field. PDFSecure address master data. Provides bidirectional and partner addresses. Overwrites *Key alias*. The *PDF Signature partner address* supports: • System variables (page 392), ${PARTNER_ADDRESS}'. • Properties (page 392), e.g. '$[PARTNER_ADDRESS]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Document password	*<Password>*[2]	Optional. Editable password field. The document owner password. Properties, system variables and XPath expressions are not supported.
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Secure password.
3) Arbitrary text.

Preconditions:

- PDFSecure Module adapter started.
- Available process attachment: PDF document expected.

Result:

- New process attachment: the verified PDF document.

Sets the following properties:

- PDF_VERIFY_RESULT := 'Document verified'.

Possible cases of failure:

Possible case of failure	Success or error behavior
PDFSecure Module adapter not started	Error. Throws acPDFSignature fault.
No certificate provided	Error. Throws acPDFSignature fault.
Digest algorithm other than 'SHA1'	Error. Throws acPDFSignature fault.

Policies: none.

1.6.3 acJasperReportGenerator

Caption: JasperReportGenerator.

Throws Fault=acJasperReportGenerator.

The *acJasperReportGenerator* activity is a single file activity that generates a report using data from multiple data sources, either the current process attachment or an SQL data source. Uses the *GenerateReport* operation of the *Jasper Reports Processor* component (refer to chapter *4.6.1 GenerateReport* of the *BIS6 Base Components* manual or the *BIC Jasper Reports Adapter* manual for more details).

There is a related activity *acPDFGenerator* (page 80), that only generates PDF reports, based on Jasper reports templates, too.

Field name	Value	Description
Report name		Mandatory. Editable text field. Default value empty. Path and name of the main Jasper report to use for report generation. The *Report name supports:* • System variables (page 392), e.g. '${REPORT_NAME}'. • Properties (page 392), e.g. '$[REPORT_NAME]'.
Input type		Mandatory. Non editable selection field. Type of source data, must match the expectations of the Jasper report in *Report name*.
	CSV	Uses the current process attachment. Expects a CSV file. Pay attention to the correct *charset* attribute set for the current process attachment. Using an incorrect encoding for data interpretation leads to corrupted characters in the generated PDF.
	SQL	Uses an SQL datasource. Makes *Datasource* visible.
	XML	Default. Uses the current process attachment. Expects an XML file. Pay attention to the correct *charset* attribute set for the current process attachment. Using an incorrect encoding for data interpretation leads to corrupted characters in the generated PDF.
Datasource	<Ref>[2]	Mandatory. The name of the SQL data source. Visible when *Input type* is set to SQL.
Output type		Mandatory. Non editable selection field. Output data format. Note that not all types fit well to the choosen *Report name*. Requirements and restrictions can be found in the Jasper Reports adapter document. An easy-to-use activity that generates PDF reports only is *acPDFGenerator* (page 80).
	DOCX	Microsoft Word® XML data format.
	ODS	Open Document Text.
	ODT	Open Document Spreadsheet.

Field name	Value	Description
	PDF	Default. Default PDF implementation generated by the Jasper Reports library.
	PDF/A-1a	PDF/A archival standard subtype 1a.
	PDF/A-1b	PDF/A archival standard subtype 1b.
	PPTX	Microsoft Powerpoint® XML data format.
	XHTML	HTML that is also well formed XML.
	XLS	Microsoft Excel® binary data format.
	XLST	Microsoft Excel® XML data format.
Locale		Optional. Editable selection field providing ENGLISH and GERMAN locales. Default value GERMAN. Can be used in the report. The *Locale* supports: • System variables (page 392), e.g. '${LOCALE}'. • Properties (page 392), e.g. '$[LOCALE]'.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

2) Reference to master data.
3) Arbitrary text.

Preconditions:

- BIS Jasper Reports Processor adapter started
- Available process attachment, if *Input type* is CSV or XML.
- Datasource, if *Input type* is SQL.

Result:

- New process attachment: the generated report in the specified output format.

Sets the following properties:

- REPORT_NAME := Entered *Report name*.
- INPUT_TYPE := Selected *Input type*.
- OUTPUT_TYPE := Selected *Output type*.
- PDFMetaData with sub properties (only available for *Output type* set to 'PDF')
  CreationDate := Date of document creation,
  PageCount := Number of PDF pages,
  Producer := Name of the report template used,
  ModDate := Date of last document modification (identical to CreationDate),
  Creator := Name of the program/library that generates the PDF.

Possible cases of failure:

Possible case of failure	Success or error behavior
BIS Jasper Reports Processor adapter not started	Error. Throws acJasperReportGenerator fault.
Empty report name	Error. Throws acJasperReportGenerator fault.
Invalid report name (typo or report does not exist)	Error. Throws acJasperReportGenerator fault.
Jasper report from *Report name* does not fit *Input type*	Error. Throws acJasperReportGenerator fault.

Policies: none.

1.6.4 ZUGFeRD

This section describes activities enabling composing and decomposing ZUGFeRD PDF files.

1.6.4.1 acZugferdCompose

Caption: ZugferdCompose.

Throws Fault=acZugferdCompose.

The *acZugferdCompose* activity is a single file activity that extracts attachments from the ZUGFeRD PDF in the current process attachment. Always expects an XML attachment that is returned as current process attachment.

Field name	Value	Description
Report name		Optional. Editable text field. Empty by default.
Report language		Optional. Editable text field. Empty by default. If this field is empty, the default language 'en' is used.
Report input		Mandatory. Editable selection field. Supports: • #
	current payload	Default. Uses the current process attachment as input file for the report generator.
	intermediatePayload	Expects a bookmarked attachment 'PAYLOAD_INTERMEDIATE'.
	partner payload	Expects a bookmarked attachment 'PAYLOAD_PARTNER'.
Report params		Optional. Editable text field. Empty by default.
Additional bookmarks to compose (name=TYPE#BOOKMARK;...)		Optional. Editable text field. Empty by default.
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.

Preconditions:

- BIC PDF/A-3 Composer/Decomposer Node adapter started.
- Available process attachment. ZUGFeRD PDF format expected.

Result:

- New process attachment: the extracted XML data file.

Sets the following properties:

- ZUGFERD_VERIFICATION_RESULT := 'OK' if the verification is switched and has succeeded
  := 'NOT_DONE' if the verification is switched off.
- ZUGFERD_ATTACHMENT_LIST := list of bookmarked PDF attachments, if bookmarking is switched on.

Possible cases of failure:

Possible case of failure	Success or error behavior
BIS PDF/A-3 Composer/Decompose adapter not started	Error. Throws acZugferdDecompose fault.
Invalid file format	Error. Throws acZugferdDecompose fault.
PDF without XML attachment	Error. Throws acZugferdDecompose fault.

Policies: none.

1.6.4.2 acZugferdDecompose

Caption: ZugferdDecompose.

Throws Fault=acZugferdDecompose.

The *acZugferdDecompose* activity is a single file activity that extracts attachments from the ZUGFeRD PDF in the current process attachment. Always expects an XML attachment that is returned as current process attachment.

Field name	Value	Description
Bookmark additional files	True	Extracts PDF attachments other than the obligatory XML attachment and bookmarks them.
	False	Default. Does not extract additional files.
Do PDF verification	True	Performs PDF verification.
	False	Default. No verification performed.
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.

Preconditions:

- BIC PDF/A-3 Composer/Decomposer Node adapter started.

- Available process attachment. ZUGFeRD PDF format expected.

Result:

- New process attachment: the extracted XML data file.
- Bookamarks any other attachment if selected.

Sets the following properties:

- ZUGFERD_VERIFICATION_RESULT := 'OK' if the verification is switched and has succeeded
  := 'NOT_DONE' if the verification is switched off.
- ZUGFERD_ATTACHMENT_LIST := list of bookmarked PDF attachments, if bookmarking is switched on.
- ZUGFERD_RESULT := 'OK' if the decomposition succeeded.

Possible cases of failure:

Possible case of failure	Success or error behavior
BIS PDF/A-3 Composer/Decompose adapter not started	Error. Throws acZugferdDecompose fault.
Invalid file format	Error. Throws acZugferdDecompose fault.
PDF without XML attachment	Error. Throws acZugferdDecompose fault.

Policies: none.

1.7 Forwarding Activities

This section describes the send activities which are not already covered by more comprehensive sections File System Activities (and), FTP Activities (), SFTP Activities (), HTTP Activities (), JMS Activities (), SLMP Activities () and SeeFX Activities ().

1.7.1 mdAS2

Caption: AS2 send.

Throws Fault=acForwardingAS2.

The *acForwardingAS2* activity is a single file activity that sends the current process attachment as an EDIINT AS2 compliant message. Uses the *sendFile* operation of the *EDIINT AS2 Adapter* component (refer to chapter *8.1.1 sendFile* of the *EDIINT AS2 Adapter* manual for more details).

Field name	Value	Description
AS2 relation		Mandatory. Editable selection field. Master data AS2 Outbound relations or Polling relations. Supports: • Properties (page 392), e.g. '$[AS2_RELATION]'. • System variables (page 392), e.g. '${AS2_RELATION}'.
	<use relation configured with ANY for own address>	Default. Makes *AS2 relation with ANY own address* below and *AS2 own address* visible.

Field name	Value	Description
	<Ref>[1]	Real master data reference of AS2 Outbound Relation or AS2 REST Partners.
AS2 relation with ANY own address	*<Ref>*[1]	Mandatory. Default value empty. Editable selection field. Master data AS2 Outbound Relations with ANY personality. Refer to chapter *7.3.3 AS2 ANY personality (link)* in the *EDIINT AS2 Adapter* manual for more details. Supports: • Properties (page 392), e.g. '$[AS2_RELATION]'. • System variables (page 392), e.g. '${AS2_RELATION}'. Since this field is editable AS2 relations with a real personality can be entered and are used the same way as the *AS2 relation* above. Note that any *AS2 own address* below is ignored in this case. *AS2 own addresses* are only used in case of ANY personality relations.
AS2 own address		Mandatory. Editable selection field. Master data AS2 personality (own address). Supports: • Properties (page 392), e.g. '$[AS2_ADDRESS]'. • System variables (page 392), e.g. '${AS2_ADDRESS}'.
	<any personality>	Default. Preinstalled default personality used as template in AS2 relations. To be replaced by real address below. Refer to chapter *7.3.3 AS2 ANY personality (link)* in the *EDIINT AS2 Adapter* manual for more details.
	<Ref>[1]	Real master data reference of AS2 personality (own address).
Subject	*<Text>*[2]	Optional. Default value empty. Editable text field. Description for this transmission. The partner can use this subject for (customer-specific) filtering. The subject of the AS2 connection is always overwritten. Supports: • Properties (page 392), e.g. '$[AS2_SUBJECT]'. • System variables (page 392), e.g. '${AS2_SUBJECT}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
File name	*<FileName>*[3]	Optional. Default value empty. Editable text field. File name used in MIME headers. If not specified, a *GUID* will be assigned automatically. Supports: • Properties (page 392), e.g. '$[AS2_FILE_NAME]'. • System variables (page 392), e.g. '${AS2_FILE_NAME}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Content type		Mandatory. Default value empty. Editable selection field. Accepts any valid MIME content-type. Provides a bunch of prefilled valid MIME content-types. Supports: • Properties (page 392), e.g. '$[AS2_CONTENTTYPE]'.

Field name	Value	Description
		• System variables (page 392), e.g. '${AS2_CONTENTTYPE}'.
	application/edifact	Default. MIME content type. EDIFACT file.
	application/edi-x12	MIME content type. EDI ANSI X12 file. Conform to specifications by the X12 standards organization.
	application/xml	MIME content type. XML file.
	text/plain	MIME content type. Plain text file.
	application/ediint-cert-exchange+xml	MIME content type. EDIINT CEM messages (certificate exchange messaging) with SMIME and XML.
	application/octet-stream	MIME content type. Files that are not specified in greater detail, e.g. executable files.
	application/pkcs-mime	MIME content type. Files according to Public Key Cryptography Standards (PKCS).
Postpone report handling	True	Does not wait for a report if it is expected (asynchronous MDN handing).
	False	Waits for an expected report.
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.
3) Valid file name.

Preconditions:

- AS2 Client adapter started.
- Available process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- REPORT_MESSAGE := 'Delivered'.
- REPORT_STATE := 'SUCCESS'.
- TP_ADAPTER := 'as2.sendAS2Message'.
- DIRECTION := 'OUT'.
- TARGET_NAME := value from *File name* field.
- TO_SUBJECT := value from *Subject* field.
- TO_NAME := 'NOT_SET'.
- TO_ID := AS2 ID of partner
- TO_UUID := UUID of AS2 outbound relation record
- AS2_REPORT_BOOKMARK := AS2 report attachment or 'NOT_SET', it not existing.
- ACTIVITY_FAULT_BOOKMARK := bookmark fault, in case of error.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid AS2 relation	Error. Throws acForwardingAS2 fault.
Invalid content type	Error. Throws acForwardingAS2 fault.
ANY personality relation with own address <any personality>	Error. Throws acForwardingAS2 fault.

Policies:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/AS2

1.7.2 mdAS4

Caption: ForwardingAS4.

Throws Fault=acForwardingAS4.

The *acForwardingAS4* activity is a single file activity that allows sending the current process attachment as an EbMS 3.0 AS4 message. Uses the *sendMessage* operation of the *AS4 Adapter* component (refer to chapter *6.2 Process* topic *sendMessage* in the *AS4 Adapter* manual for more details).

Mode of operation:
Sends a User Message containing the current process attachment as payload to the partner MSH (Messaging Services Handler). Supports one-way/push and one-way/pull exchange patterns.

Field name	Value	Description
AS4 agreement	*<Ref>*[1]	Mandatory. Editable selection field. Master data AS4 sending agreements or bidirectional agreements. The *AS4 agreement* supports: • Properties (page 410), e.g. '$[AS4_AGREEMENT]'. • System variables (page 392), e.g. '${AS4_AGREEMENT}'.
Collaboration information service name	*<Text>*[2]	Optional. Editable text field. The service name of the collaboration information (business context) i.e. the "Service" value in the message. It overwrites the settings made in the "BusinessInformation" tab of the *AS4 agreement*. The *Collaboration information service name* supports: • Properties (page 410), e.g. '$[SERVICE_NAME]'. • System variables (page 392), e.g. '${SERVICE_NAME}'. • XPath expressions (page 392), e.g. $(upper-case('serviceName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Collaboration information action	*<Text>*[2]	Optional. Editable text field. The action of the collaboration information (business context) i.e. the "Action" value in the message. It overwrites the settings made in the "BusinessInformation" tab of the *AS4 agreement*. The *Collaboration information service action* supports: • Properties (page 410), e.g. '$[SERVICE_ACTION]'. • System variables (page 392), e.g. '${SERVICE_ACTION}'.

Field name	Value	Description
		• XPath expressions (page 392), e.g. $(upper-case('action')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Extended sender and receiver settings	True	Provide additional AS4 settings concerning party IDs, information types, and information roles. Makes the following fields visible: *Sender information party ID*, *Sender information type*, *Sender information role*, *Receiver information party ID*, *Receiver information type*, and *Receiver information role*.
	False	Default. Do not provide any additional settings.
Sender information party ID	*<Text>*[2]	Mandatory. Editable text field. Empty by default. AS4 party ID for the sender. It overwrites the settings made in the "Party IDs" tab of the own address set in *AS4 agreement*. The *Sender information party ID* supports: • Properties (page 410), e.g. '$[PARTY_ID]'. • System variables (page 392), e.g. '${PARTY_ID}'. • XPath expressions (page 392), e.g. $(upper-case('myID')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended sender and receiver settings* is switched on.
Sender information type	*<Text>*[2]	Mandatory. Editable text field. Empty by default. Type of the AS4 party ID for the sender. It overwrites the settings made in the "Party IDs" tab of the own address set in *AS4 agreement*. The *Sender information type* supports: • Properties (page 410), e.g. '$[TYPE]'. • System variables (page 392), e.g. '${TYPE}'. • XPath expressions (page 392), e.g. $(upper-case('myType')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended sender and receiver settings* is switched on.
Sender information role	*<Text>*[2]	Mandatory. Editable text field. Empty by default. Role of the sender set in the message. It overwrites the settings made in the "BusinessInformation" tab of the *AS4 agreement*. The *Sender information role* supports: • Properties (page 410), e.g. '$[ROLE]'. • System variables (page 392), e.g. '${ROLE}'. • XPath expressions (page 392), e.g. $(upper-case('myRole')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended sender and receiver settings* is switched on.
Receiver information party ID	*<Text>*[2]	Mandatory. Editable text field. Empty by default. AS4 party ID for the receiver. It overwrites the settings made in the "Party IDs" tab of the partner address set in *AS4 agreement*. The *Receiver information party ID* supports:

Field name	Value	Description
		- Properties (page 410), e.g. '$[PARTY_ID]'. - System variables (page 392), e.g. '${PARTY_ID}'. - XPath expressions (page 392), e.g. $(upper-case('myID')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended sender and receiver settings* is switched on.
Receiver information type	*<Text>*[2]	Mandatory. Editable text field. Empty by default. Type of the AS4 party ID for the receiver. It overwrites the settings made in the "Party IDs" tab of the partner address set in *AS4 agreement*. The *Receiver information type* supports: - Properties (page 410) , e.g. '$[TYPE]'. - System variables (page 392), e.g. '${TYPE}'. - XPath expressions (page 392), e.g. $(upper-case('myType')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended sender and receiver settings* is switched on.
Receiver information role	*<Text>*[2]	Mandatory. Editable text field. Empty by default. Role of hte receiver set in the message. It overwrites the settings made in the "BusinessInformation" tab of the *AS4 agreement*. The *Receiver information role* supports: - Properties (page 410), e.g. '$[ROLE]'. - System variables (page 392), e.g. '${ROLE}'. - XPath expressions (page 392), e.g. $(upper-case('myRole')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended sender and receiver settings* is switched on.
Content type		Mandatory. Editable selection mode. MIME type for the attachment. Provides a bunch of prefilled valid MIME content-types. Accepts any valid MIME content-type. Properties and system variables are not supported.
	application/edifact	Default. MIME content-type. EDIFACT file.
	application/edi-x12	MIME content-type. EDI ANSI X12 file. Conform to specifications by the X12 standards organization.
	application/xml	MIME content-type. XML file.
	text/plain	MIME content-type. Plain text file.
	text/html	MIME content-type.HTML file.
	text/richtext	MIME content-type. Richtext file.
	application/ediint-cert-exchange+html	MIME content-type. EDIINT CEM messages (certificate exchange messaging) with SMIME and XML.
	application/octet-stream	MIME content-type. Files that are not specified in greater detail, e.g. executable files.

Field name	Value	Description
	application/pkcs7-mime	MIME content-type. Files according to Public Key Cryptography Standards (PKCS).
Part properties(property=value;...;)	*<KVList>*[3]	Optional. Editable text field. Empty by default. Property list for the attachment. Maximum count is 100. The *Part properties* supports: • Properties (page 410), e.g. '$[PROPS]'. • System variables (page 392), e.g. '${PROPS}'. • XPath expressions (page 392), e.g. $(upper-case('myRole')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3)

1) Reference to master data.
2) Arbitrary text.

Preconditions:

- AS4 Client adapter started.
- Available process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- RESULT := 'Sent'.
- STATUS := 'SUCCESS_NOTIFY'.
- TO_ADAPTER := 'as4.sendAS4Message'.
- DIRECTION := 'OUT'.
- CORRELATION_ID := <used ID for correlation>.
- CONVERSATION_ID := <used ID>.

Possible cases of failure:

Possible case of failure	Success or error behavior
AS4 Client adapter not started.	Error. Throws acForwardingAS4 fault.
Invalid system variable in editable field	Error. Throws acForwardingAS4 fault.
Invalid property in editable field	Error. Throws acForwardingAS4 fault.
Invalid MIME content type	Success. Sends the process attachment.

Policies: none.

1.7.3 mdASP

Caption: ASP connect.

Throws Fault=acForwardingASP.

⚠ **Internal use only!**

The *mdASP* activity is a single file activity that sends the current process attachment via ASP. Uses the *sendMessage* operation of the *SLMP Adapter* component (refer to chapter *8.1.1 sendMessage (Sending Messages))* in the *SLMP Adapter* manual for more details).

Field name	Value	Description
Own ID	*<Ref>*[1]	Mandatory. Non editable selection field. Master data SLMP Own address or bidirectional address.
Receiver ID	*<Ref>*[1]	Optional. Editable selection field. Empty by default. Master data SLMP Partner address or bidirectional address.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3)

1) Reference to master data.
2) Arbitrary text.

Policy: none.

1.7.4 mdMAIL

Caption: MAIL send.

Throws Fault=acForwardingMAIL.

The *mdMAIL* activity is a multiple file activity that can send several attachments such as the current process attachment or bookmarks. Uses the sendMail operation from the Mail Client component (refer to chapter *8.1 sendMail (Sending Mail Messages) (link)* of the *Mail Adapter* manual for more details).

Field name	Value	Description
Sender	*<Ref>*[1]	Mandatory. Editable selection field. Master data Mail Own Addresses or bidirectional Addresses. Properties, system variables and email addresses are not supported.
Receiver	*<Ref>*[1]	Mandatory. Editable selection field. Master data Mail Partner Addresses or bidirectional Addresses provided by drop-down list. Receiver email addresses or list of email addresses can be also entered. The *Receiver* supports: • Properties (page 392), e.g. '$[EMAIL]'. • Email address and semicolon separated email address list. System variables are not supported.

Field name	Value	Description
Receiver cc	*<CCList>*[2]	Optional. Empty by default. Editable text field. List of additional receiver email addresses (CC). The *Receiver cc* supports: • Email address and semicolon separated email address list. • Properties (page 392), e.g. '$[EMAIL]'. • System variables (page 392), e.g. '${EMAIL}'. • Maximum number of addresses is 2000. Additional addresses are ignored.
Receiver bcc	*<CCList>*[2]	Optional. Empty by default. Editable text field. List of additional receiver email addresses (BCC). The *Receiver bcc* supports: • Email address and semicolon separated email address list. • Maximum number of addresses is 2000. Additional addresses are ignored. Properties and system variables are not supported.
Connection	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides available master data mail connections.
Subject	*<Text>*[3]	Optional. Default value '$[FROM_SUBJECT]'. Editable text field. The mail message’s subject. The *Subject* supports: • Properties (page 392), e.g. '$[MAIL_SUBJECT]'. • System variables (page 392), e.g. '${MAIL_SUBJECT}'.
Send format		Mandatory. Non-editable selection field. Provides several mail message formats.
	Send text	Default. Send a mail message with a message body and without any further attachments. The body is created by either entering the text directly or by accessing a bookmarked attachment. Makes *Take body from a bookmark* visible.
	Send file as attachment	Send a mail message with a message body and with the current process attachment as attachment with target name *File name*. The body is created by either entering the text directly or by accessing a bookmarked attachment. Makes *File name* and *Take body from a bookmark* visible.
	Send bookmarks as attachment	Send a mail message with a message body and with a list of bookmarks from *Add attachments from bookmarks* as attachments. The body is created by either entering the text directly or by accessing a bookmarked attachment. Makes *Add attachments from bookmarks* and *Take body from a bookmark* visible.
	Send file as body	Send a mail message with the current process attachment as mail message body and without any further attachments.
File name	*<Text>*[3]	Optional. Default value '$[FROM_SUBJECT]'. Editable text field. Visible when *Send format* set to 'Send file as attachment'. Name of the mail message attachment. If *File name* is empty, the process ID is used as file name, for instance '10c3ffd0-0c7b-11e6-ab0d-f5dd0a0069b0.txt'.
Add attachments from bookmarks	*<Bookmark* List>[4]	Mandatory. Empty by default. Editable text field. Visible when *Send format* set to 'Send bookmarks as attachment'. Sends the specified bookmarks as mail message attachments. The *Add attachments form bookmarks* supports:

Field name	Value	Description
		• Properties (page 392), e.g. '$[MAIN_BM];$[STD_BM];ATTACHMENT_1'. • System variables (page 392), e.g. '${MAIN_BM};${STD_BM};ATTACHMENT_1'. • Maximum number of bookmarks is 2000. Additional bookmarks are ignored.
Take body from a bookmark		Mandatory. Visible when *Send format* set to 'Send text', 'Send file as attachment', or 'Send bookmarks as attachment'.
	True	Create a mail message body from *Body bookmark*.
	False	Default. Create a mail message body from *Body text*.
Content type of body		Mandatory. Editable selection field. MIME content content type for the mail body. Properties and system variables are not supported.
	application/msexcel	MIME content-type. Microsoft Excel file.
	application/msword	MIME content-type. Microsoft Word file.
	application/octet-stream	MIME content-type. Files that are not specified in greater detail, e.g. executable files.
	application/gzip	MIME content-type. GNU zip file.
	text/plain	Default. MIME content-type. Plain text file.
	text/CSV	MIME content-type. CSV file.
	text/html	MIME content-type. HTML file.
	text/richtext	MIME content-type. Richtext file.
	text/rtf	MIME content-type. RTF file.
	image/jpeg	MIME content-type. JPEG file.
	image/tiff	MIME content-type. TIFF file.
Body text	*<Text>*[3]	Optional. Default value 'this is the body'. Editable text field. Visible when *Send format* set to 'Send text', 'Send file as attachment', of 'Send bookmarks as attachment' and *Take body from a bookmark* is switched off. The *Body text* supports: • Properties (page 392), e.g. '$[MY_MAIL_BODY]'. • System variables (page 392), e.g. '${MAIL_BODY}'. • Escaped XML. • XPath expressions (page 392), e.g. $(upper-case('boundary')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Body bookmark	*<Bookmark>*[5]	Optional. Default value empty. Editable text field. Visible when *S end format* set to 'Send text', 'Send file as attachment', of 'Send bookmarks as attachment' and *Take body from a bookmark* is switched on. The Body bookmark supports: • Properties (page 392), e.g. '$[MAIL_BODY_BOOKMARK]'.

Field name	Value	Description
		- System variables (page 392), e.g. '${MAIL_BODY_BOOKMARK}'.
Priority		Optional. Empty by default. Selection field. Provides 3 priority levels.
	High	High priority.
	Normal	Standard priority.
	Low	Low priority.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Semicolon separated list of receiver addresses. Either names of master data Mail Addresses or valid Email addresses.
3) Arbitrary text.
4) Semicolon separated list of case-insensitive bookmark names.
5) Existing bookmark.

Preconditions:

- Mail Client adapter started.
- Available process attachment in case of *Send format* set to 'Send file as attachment' or 'Send file as body'.
- Specified booksmarks in case of *Send format* set to 'Send text' or 'Send bookmarks as attachment'.
- Specified bookmark in case *Take body from a bookmark* is switched on.

Result:

- Current process attachment: not changed.

Sets the following properties:

- TO_UUID := UUID of receiver address. 'NOT_SET' if it an email address.
- TO_SUBJECT := Subject of the mail. sFwdSubject, 'NOT_SET' if no subject is entered.
- TO_ADAPTER := 'dtMail.sendMail'.
- TO_ID := 'NOT_SET'.
- TO_NAME := semicolon separated list of email recipients from Receiver, Receiver cc, and Receiver bcc fields (only those containing '@'), e.g. 'to:{user1@domain};cc:{user2@domain};bcc:{user3@domain}'. If Receiver does not contain a '@', 'NOT_SET' is used no matter what the other recipients look like, e.g. 'NOT_SET;cc:{user@domain}'.
- DIRECTION := 'OUT'.
- TARGET_NAME := File name, if needed and given depending on Send format; 'NOT_SET' otherwise.
- RESULT := 'Sent'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid email address format in Receiver * fields	Error. Throws acForwardingX400P7 fault.
Property in Content type of body	Error. Throws acForwardingX400P7 fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/MAIL

1.7.5 mdOFTP

Caption: OFTP send.

Throws Fault=acForwardingOFTP.

The *mdOFTP* activity is a single file activity that allows sending the current process attachment as an OFTP message. Uses the *sendFile* operation from the *OFTP Adapter* component (refer to chapter *8.2 sendFile (Sending Files) (link)* of the OFTP Adapter manual for more details).

Field name	Value	Description
Receiver address	*<Ref>*[1]	Mandatory. Editable selection field. Master data OFTP client partner address or bidirectional address. OFTP message recipient. Must match the Own Address of Session partners of selected *Connection*. The *Receiver address* supports: • Properties (page 392), e.g. '$[OWN_ADDRESS]'. • System variables (page 392), e.g. '${OWN_ADDRESS}'. The *Receiver address* has to match the *Connection* Own address otherwise this activity will fail.
Sender address	*<Ref>*[1]	Mandatory. Editable selection field. Master data OFTP client own address or bidirectional address. OFTP message sender. Must match the Partner Address of Session partners of selected *Connection*. The *Receiver address* supports: • Properties (page 392), e.g. '$[PARTNER_ADDRESS]'. • System variables (page 392), e.g. '${PARTNER_ADDRESS}'. The *Sender address* has to match the *Connection* Partner address otherwise this activity will fail.
Connection	*<Ref>*[1]	Mandatory. Editable selection field. Master data OFTP client connection, either ISDN, TCP or X.31. Holds the OFTP session data. The *Receiver address* supports: • Properties (page 392), e.g. '$[CONNECTION]'. • System variables (page 392), e.g. '${CONNECTION}'. The Own address and the partner address of this OFTP client connection have to match *Receiver address* and *Sender address* otherwise this activity will fail.
Logical filename	*<Text>*[2]	Mandatory. Editable text field. The logical file name for the transfer. The *Logical filename* supports: • Properties (page 392), e.g. '$[LOG_FILENAME]'. • System variables (page 392), e.g. '${LOG_FILENAME}'.

Field name	Value	Description
		• XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Send format		Mandatory. Editable selection field. Provides the supported file content types. The *Send format* supports: • Properties (page 392), e.g. '$[SEND_FORMAT]'. System variables are not supported.
	UNSTRUCTURED	Default. Binary type.
	VARIABLE	Variable record length. Makes *Max. record size* visible.
	FIX	Fixed record length. Makes *Record size* visible.
	TEXT	ASCII text.
Max. record size	*<Int>*[3]	Optional. Editable text field. Default value 128. Maximally allowed value for the record length. The *Max. record size* supports: • Properties (page 392), e.g. '$[RECORD_SIZE]'. • System variables (page 392), e.g. '${RECORD_SIZE}'. Visible when *Send format* is set to 'Variable'.
Record size	*<Int>*[3]	Optional. Editable text field. Default value 128. Maximally allowed value for the record length. The *Max. record size* supports: • Properties (page 392), e.g. '$[RECORD_SIZE]'. • System variables (page 392), e.g. '${RECORD_SIZE}'. Visible when *Send format* is set to 'Fix'.
V2-description	*<Text>*[2]	Optional. Editable text field. Virtual file description (v2SFIDSettings). The *V2-description* supports: • Properties (page 392), e.g. '$[V2_DESCRIPTION]'. • System variables (page 392), e.g. '${V2_DESCRIPTION}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.
3) Positive Integer number.

Preconditions:

- OFTP-ISDN, OFTP-TCP or OFTP-X31 Client adapter started depending on type of *Connection*.

- Available process attachment.

Result:

- Current process attachment: not changed.
- DT report as bookmarked attachment: UTF-8 character set used.

Sets the following properties:

- OFTP_REPORT_BOOKMARK := bookmarked attachment containing DT report.
- REPORT_STATE := 'SUCCESS' in case of success.
- TARGET_NAME := Logical file name.
- DIRECTION := 'OUT'.
- TO_NAME := 'NOT_SET'.
- TO_ID := 'NOT_SET'.
- TO_SUBJECT := Logical file name.
- TO_UUID := UUDI of receiver address.
- TO_ADAPTER := 'oftp.sendFile'.
- REPORT_MESSAGE := sReportMessage

Possible cases of failure:

Possible case of failure	Success or error behavior
Not matching *Receiver address* and *Connection Own address*	Error. Throws acForwardingOFTP fault.
Not matching *Sender address* and *Connection Partner address*	Error. Throws acForwardingOFTP fault.
Invalid *Send format*	Error. Throws acForwardingOFTP fault.
Invalid *Record size*	Error. Throws acForwardingOFTP fault.
Invalid *Max. record size*	Error. Throws acForwardingOFTP fault.

Policy: none.

1.7.6 mdX400

Caption: X400 send.

Throws Fault=acForwardingX400P7.

The *acForwardingX400P7* activity is a single file activity that allows sending the current process attachment using the X.400 mail system and a variety of message types (i.e. bytes, message, text). Uses the *sendFileMD* operation from the *X400 P7 Adapter* component (refer to chapter *8.2 sendFileMD (Sending Messages) (link)* of the X.400-P7 Client manual for more details).

Field name	Value	Description
Receiver	*<Ref>*[1]	Mandatory. Editable selection field. Master data X.400-P7 partner address or bidirectional address. The *Receiver* supports: • Properties (page 392), e.g. '$[X400_RECEIVER]'. • System variables (page 392), e.g. '${X400_RECEIVER}'.

Field name	Value	Description
		• XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Sender	*<Ref>*[1]	Mandatory. Editable selection field. Master data X.400-P7 own address or bidirectional address. The *Sender* supports: • Properties (page 392), e.g. '$[X400_SENDER]'. • System variables (page 392), e.g. '${X400_SENDER}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Connection	*<Ref>*[1]	Mandatory. Editable selection field. Master data X.400-P7 connection. The *Connection* supports: • Properties (page 392), e.g. '$[X400_CONNECTION]'. • System variables (page 392), e.g. '${X400_CONNECTION}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
File name	*<Text>*[2]	Mandatory. Editable text field. Empty by default. Short message description. The *File name* supports: • Properties (page 392), e.g. '$[X400_FILE_NAME]'. • System variables (page 392), e.g. '${X400_FILE_NAME}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Send format		Mandatory. Non editable selection field. Empty by default. Provides supported send formats i.e. specifications of the structure of the message content.
	TEXT	IA5 text (International Reference Alphabet No. 5).
	ATTACHMENT	Bilaterally defined.
	Eight-Bit-ASCII	General text.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.

Preconditions:

- X.400-P7 Client adapter started.

- Available process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- TARGET_NAME := File name.
- TARGET_PATH := 'NOT_SET'.
- X400_CORRELATION_ID := generated correlation ID.
- DIRECTION := 'OUT'.
- TO_NAME := 'NOT_SET'.
- TO_ADAPTER := 'x400.sendFileMD'.
- REPORT_STATE := DT report state.
- TO_UUID := UUID of Receiver Address.
- TO_SUBJECT := File name.
- TO_ID := 'NOT_SET'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid property in editable field	Error. Throws acForwardingX400P7 fault.
Invalid system variable in editable field	Error. Throws acForwardingX400P7 fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/JMS ALLOWANCE/SYSTEM-EXECUTE-SCRIPT

1.8 File System Activities

This section describes file system related activities.

1.8.1 acFileClientList

Caption: FileClientList.

Throws Fault=acFileClientList.

The *acFileClientList* activity lists multiple files of a directory or directory tree and returns a FILELIST object (CSV file) that can be used as input to the *scForEach* (page 10) activity providing a loop element. It uses the *listDirectory* operation from the *FileClient Adapter* component (refer to chapter *1.11.4 listDirectory (Lists contents of a directory))* of the *BIS6 Base Components* manual or chapter *8.7 listDirectory (Lists contents of a directory)* of the *File System Adapter* manual for more details).

Mode of operation:
The *FileClient Adapter* component lists the directory for files matching the file mask and returns a list of file names and file paths. This list is converted into CSV format by the *BIC Client* adapter and can be used as input for the *scForEach* (page 10) looping activity.

Field name	Value	Description
File mask	*<Pattern>*[1]	Mandatory. Editable text field. Default value is '*.*'. Case sensitive file list filter acting as include pattern. Processes only files that match this condition. The *File mask* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. • Character sets matching exactly one character of the specified set, e.g. '[abc]'. • Character ranges matching exactly one character of the specified range, e.g. '[a-z]'. • Relative paths where the relative path is related to the *Connection* root path. The path part does not support any mask patterns. Subdirectories or a complete directory tree can be included by setting *Recursive search depth into folder*. • System variables (page 392), e.g. '${MY_INCLUDE_PATTERN}'. • Properties (page 392), e.g. '$[MY_INCLUDE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Exclude pattern	*<Pattern>*[1]	Optional. Editable text field. Empty by default. Case sensitive exclusion file mask. Processes only files that do not match this condition. The *Exclude pattern* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. • Character sets matching exactly one character of the specified set, e.g. '[abc]'. • Character ranges matching exactly one character of the specified range, e.g. '[a-z]'. • System variables (page 392), e.g. '${MY_EXCLUDE_PATTERN}'. • Properties (page 392), e.g. '$[MY_EXCLUDE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Relative paths are not supported.
Recursive search depth into folder	*<PosInt>*[2]	Mandatory. Editable number filed. Default value '0'. Determines the level in the directory tree, starting from the listed directory, in which matching files will be searched. • 0: Default value. Inspect only the root folder, i.e. the directory specified by *Connection* and *Address*. • >0: The subdirectory level that is to be scanned in addition providing a recursive search.
Mapping parameter		Optional. Editable text field. Empty by default. Mapping parameter for BIC conversion into CSV format. Available for the mapping as an AdditionalParameter with the name 'PARAMETER'. The *Mapping parameter* supports:

Field name	Value	Description
		• System variables (page 392), e.g. '${MY_MAPPING_PARAM}'. • Properties (page 392), e.g. '$[MY_MAPPING_PARAM]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000'))
File system type		Mandatory. Non-editable selection field. Type of file system resources to use. Affects the lists of available addresses and connections below.
	Local	Default. Local file resources.
	CIFS	CIFS shared folders on file storage resources that support the CIFS protocol using share names.
Address	*<Ref>*[3]	Mandatory. Editable selection field. Master data file client address according to the *File system type*. The relative path to the directory to be listed, in relation to the root path. The *Address* supports: • System variables (page 392), e.g. '${ADDRESS}'. • Properties (page 392), e.g. '$[ADDRESS]'.
Connection	*<Ref>*[3]	Mandatory. Editable selection field. Master data file client connection according the *File system type*. Denotes the root path, i.e. the absolute path name of a directory or for local access, or the share name (or directory starting with the share name) for CIFS access. The *Connection* supports: • System variables (page 392), e.g. '${CONNECTION}'. • Properties (page 392), e.g. '$[CONNECTION]'.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[3]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Case sensitive file mask pattern allowing wildcards.
2) Positive integer number.
3) Reference to master data.
4) Arbitrary text.

Additional configuration:

System variable	Description
MFT-FILELISTCREATOR	Name of mapping for conversion of file list into FILELIST CSV format. ⚠ Changing this mapping affects all list activities with FILELIST format!

Preconditions:

- File Client adapter started.
- BIC Client adapter started.
- Mapping from ${MFT-FILELISTCREATOR} system variable available in repository.

Result:

- New process attachment: CSV file containing the file list.

Sets the following properties:

- LIST_COUNT := Number of entries in the returned file list.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acFileClientList fault.
Directory without permission	Success. Returns a list with 0 files.
Empty directory	Success. Returns a list with 0 files.
File without permission	Success. Returns a list with all files.
Locked file	Success. Returns a list with all files.
Not existing *Connection*	Error. Throws acFileClientList fault.
Not existing *Address*	Error. Throws acFileClientList fault.
CIFS with wrong host name	Error. Throws acFileClientList fault.
CIFS with wrong share name	Error. Throws acFileClientList fault.
CIFS with wrong authentication	Error. Throws acFileClientList fault.
CIFS with wrong domain name	Error. Throws acFileClientList fault.

Policy: none.

1.8.2 acFileClientExists

Caption: FileClientExists.

Throws Fault=acFileClientExists.

The *acFileClientList* activity is an activity that checks the existence of a directory or a file. It uses the *listDirectory* operation from the *FileClient Adapter* component (refer to chapter *1.11.4 listDirectory (Lists contents of a directory)* of the *BIS6 Base Components* manual or chapter *8.7 listDirectory (Lists contents of a directory)* of the *FileSystem Adapter* manual for more details).

Field name	Value	Description
Mask	*<Pattern>*[1]	Mandatory. Editable text field. Default value is '*.*'. Handling depends on *Object type* below. Checks for existence of a folder specified by its name or existence of at least one file matching a file mask. The *Mask* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character (Note: for 'File' *Object type* only). • Character sets matching exactly one character of the specified set, e.g. '[abc]' (Note: for 'File' *Object type* only). • Character ranges matching exactly one character of the specified range, e.g. '[a-z]' (Note: for 'File' *Object type* only). • Relative paths where the relative path is related to the *Connection* root path. The path part does not support any mask patterns. • System variables (page 392), e.g. '${MY_PATTERN}'.

Field name	Value	Description
		- Properties (page 392), e.g. '$[MY_PATTERN]'. - XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Object type		Mandatory. Non-editable selection field. Switch between checking files and folders.
	File	Default. Checks the existence of a file.
	Folder	Checks the existence of a directory.
File system type		Mandatory. Non-editable selection field. Type of file system resources to use. Affects the lists of available addresses and connections below.
	CIFS	CIFS shared folders on file storage resources that support the CIFS protocol using share names.
	local	Default. Local file resources.
Address	*<Ref>*[2]	Mandatory. Editable selection field. Master data file client address according to the *File system type*. The relative path to the directory to be listed, in relation to the root path. The *Address* supports: - System variables (page 392), e.g. '${ADDRESS}'. - Properties (page 392), e.g. '$[ADDRESS]'.
Connection	*<Ref>*[2]	Mandatory. Editable selection field. Master data file client connection according the *File system type*. Denotes the root path, i.e. the absolute path name of a directory or for local access, or the share name (or directory starting with the share name) for CIFS access. The *Connection* supports: - System variables (page 392), e.g. '${CONNECTION}'. - Properties (page 392), e.g. '$[CONNECTION]'.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Case sensitive file mask pattern allowing wildcards.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- File Client adapter started.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- FILE_OBJECT_FOUND := 'TRUE', if file/directory was found.
  := 'FALSE', if file/directory was not found.
- TARGET_NAME := *File mask* value in case of 'File' *Object type*, empty in case of 'Folder' *Object type*.

- TARGET_PATH := name of the listed directory.
- OBJECT_TYPE := *Object type* value.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Object type	Success or error behavior
Not existing directory		Success. FILE_OBJECT_FOUND := 'FALSE'.
Directory without permission	File	Success. FILE_OBJECT_FOUND := 'FALSE'.
Directory without permission	Directory	Success. FILE_OBJECT_FOUND := 'TRUE'.
Empty directory	File	Success. FILE_OBJECT_FOUND := 'FALSE'.
Empty directory	Directory	Success. FILE_OBJECT_FOUND := 'TRUE'.
File without permission	File	Success. FILE_OBJECT_FOUND := 'TRUE'.
Locked file	File	Success. FILE_OBJECT_FOUND := 'TRUE'.
Not existing *Connection*		Error. Throws acFileClientExists fault.
Not existing *Address*		Error. Throws acFileClientExists fault.
CIFS with wrong host name		Error. Throws acFileClientExists fault.
CIFS with wrong share name		Error. Throws acFileClientExists fault.
CIFS with wrong authentication		Error. Throws acFileClientExists fault.
CIFS with wrong domain name		Error. Throws acFileClientExists fault.

Policy: none.

1.8.3 acFileClientPickFile

Caption: Pick file.

Throws Fault=acFileClientPickFile.

The *acFileClientPickFile* activity is a single file activity that picks up a file from a local file system or a CIFS shared folder. It uses the *getFile* operation from the *FileSystem Adapter* component (refer to chapter *1.11.2* *getFile (Retrieves file as attachment)* of the *BIS6 Base Components* manual or chapter *8.3 getFile (Retrieves file as attachment)* of the *FileSystem Adapter* manual for more details).

File path	*<FilePath>*[1]	Mandatory. Editable text field. The name of the source file. Supports: • Relative path to the parent directory of the file, in relation to the *C onnection* root path. This relative path overwrites the relative path defined by *Address*. • System variables (page 392), e.g. '${RELATIVE_PATH}*.dta'. • Properties (page 392), e.g. '$[RELATIVE_PATH]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
File system type		Mandatory. Non-editable selection field. Type of file system resources to use. Affects the lists of available *Addresses* and *Conne ctions* below.
	Local	Default. Local file resources.
	CIFS	CIFS shared folders on file storage resources that support the CIFS protocol using share names.
Address	*<Ref>*[2]	Mandatory. Editable selection field. Master data file client address according to the *File system type*. The relative path to the directory in relation to the *Connection* root path. The *Address* supports: • System variables (page 392), e.g. '${ADDRESS}'. • Properties (page 392), e.g. '$[ADDRESS]'.
Connection	*<Ref>*[2]	Mandatory. Editable selection field. Master data file client connection according to the *File system type*. Denotes the root path, i.e. the absolute path name of a directory for local access, or the share name (or directory starting with the share name) for CIFS access. The *Connection* supports: • System variables (page 392), e.g. '${CONNECTION}'. • Properties (page 392), e.g. '$[CONNECTION]'.
Delete picked file	True	Delete the source file.
	False	Default. Keep the source file.
Retry if not found	True	Retry this activity in cases of failure in compliance with the *Retry co unt* and *Retry delay* parameters below. This type of retry takes place additionally to the retry provided by the *Queue*.
	False	Default. Do not retry in cases of failure. This type of retry does not affect the retry provided by the *Queue*.
Retry count	*<PosInt>*[3]	Default value: 3. The maximum number of retries for this activity. Minimum values is 1. Only taken into account if *Retry if not found* is switched on.
Retry delay in seconds	*<PosInt>*[3]	Default value: 30. The time in seconds between two retries for processing this activity. Only taken into account if *Retry if not found* is switched on.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).

Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Valid relative file path and valid file name.
2) Reference to master data.
3) Positive integer.
4) Arbitrary text.

Preconditions:

- File client adapter started.

Result:

- New process attachment: Picked file.

Sets the following properties:

- TARGET_NAME := The name of the target file.
- TARGET_PATH := The path of the target file.
- FILE_COUNT := The number of picked files, 1 in case of real success, 0 in case of error when the *Abort on error* flag is not switched on.
- DIRECTION := 'IN'.
- STATUS := 'SUCCESS_NOTIFY'.
- RESULT := 'Files picked up'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acFileClientPickFile fault.
Empty directory.	Error. Throws acFileClientPickFile fault.
Directory without permission	Error. Throws acFileClientPickFile fault.
Not existing file	Error. Throws acFileClientPickFile fault.
File path references directory	Error. Throws acFileClientPickFile fault.
Empty file	Success. File is picked up.
File without permission	Error. Throws acFileClientPickFile fault.
Locked file	Error. Throws acFileClientPickFile fault.
File name with unsupported characters	Error. Throws acFileClientPickFile fault.
Not existing *Connection*	Error. Throws acFileClientPickFile fault.
Not existing *Address*	Error. Throws acFileClientPickFile fault.
CIFS with wrong host name	Error. Throws acFileClientPickFile fault.
CIFS with wrong share name	Error. Throws acFileClientPickFile fault.
CIFS with wrong authentication	Error. Throws acFileClientPickFile fault.
CIFS with wrong domain name	Error. Throws acFileClientPickFile fault.

Policy: none.

1.8.4 acFileClientPolling

Caption: FileClientPolling.

Throws Fault=acFileClientPolling.

The *acFileClientPolling* activity polls multiple files from a local directory or a CIFS shared folder that trigger one or more new MFT-FileSystemEvents (refer to chapter *MFT-FileSystemEvents* in the *MFT Standard Solution - Adapters* manual for more details). It uses the *polling* operation from the *FileSystem Adapter* component (refer to chapter *1.11.5 polling (Polls a directory))* of the *BIS6 Base Components* manual or chapter *8.8 polling (Polls a directory)* of the *FileSystem Adapter* manual for more details).

File mask	*<Pattern>*[1]	Mandatory. Editable text field. Default value '*.*'. Case sensitive file list filter acting as include pattern. Files matching to this mask will be polled. The *File mask* supports • Wildcard characters '*' - representing zero or more symbols, and '?' - representing exactly one symbol. • Relative paths where the relative path is related to the *Connecti on* root path. The *Address* relative path is overwritten. The path part does not support any mask patterns. Subdirectories are not included. • System variables (page 392), e.g. '${MY_INCLUDE_PATTERN}*.dta'. • Properties (page 392), e.g. '$[MY_INCLUDE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000'))
Exclude pattern	*<Pattern>*[1]	Optional. Editable text field. Empty by default. Case sensitive exclusion file mask. Files matching to this mask will not be polled. The *Exclude pattern* supports • Wildcard characters '*' - representing zero or more symbols, and '?' - representing exactly one symbol. • System variables (page 392), e.g. '${MY_EXCLUDE_PATTERN}'. • Properties (page 392), e.g. '$[MY_EXCLUDE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')) Relative paths are not supported.
File system type		Mandatory. Non-editable selection field. Type of file system resources to use. Affects the lists of available *Addresses* and *Conne ctions* below.
	Local	Default. Local file resources.
	CIFS	CIFS shared folders on file storage resources that support the CIFS protocol using share names.
Address	*<Ref>*[2]	Mandatory. Editable selection field. Master data File Client address according to the *file system type*.

		The relative path to the directory to be listed, in relation to the *Connection* root path. The *Address* supports: • System variables (page 392), e.g. '${ADDRESS}'. • Properties (page 392), e.g. '$[ADDRESS]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Connection	*<Ref>*[2]	Mandatory. Editable selection field. Master data File Client connection according to the *file system type*. Means the root path: The absolute path name of a directory for local access, or the share name (or directory path starting with the share name) for CIFS access. The *Connection* supports: • System variables (page 392), e.g. '${CONNECTION}'. • Properties (page 392), e.g. '$[CONNECTION]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Delete polled files	True	Delete the files that have been polled.
	False	Default. Keep the files that have been polled.
Append polled files	True	All matching files will be appended to a single attachment. Exactly one MFT-FileSystemEvent is triggered.
	False	Default. Each polled file is put into a separate attachment of its own. Each polled file triggers a new MFT-FileSystemEvent.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Case sensitive file mask pattern allowing wildcards.
2) Reference to master.
3) Arbitrary text.

Preconditions:

- File client adapter started.

Result:

- New MFT-FileSystemEvents for the polled files.
- Current process attachent: not accessed and not changed.

Sets the following properties:

- FILE_COUNT := The number of polled files. If an error is raised, it is set to 0.
- FILES_LEFT := The number of files that have not been polled.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acFileClientPickFile fault.
Empty directory	Success. No files are polled. FILE_COUNT is set to 0. FILES_LEFT is set to 0.
Directory without permission	Success. No files are polled. FILE_COUNT is set to 0. FILES_LEFT is set to 0.
Not existing file/s	Success. No files are polled. FILE_COUNT is set to 0. FILES_LEFT is set to 0.
Empty file	Success. File is polled. FILE_COUNT is increased. FILES_LEFT is not increased.
File without permission	Success. File is not polled. FILE_COUNT is not increased. FILES_LEFT is increased.
Locked file	Success. File is not polled. FILE_COUNT is not increased. FILES_LEFT is increased.
Not existing *Connection*	Error. Throws acFileClientPickFile fault.
Not existing *Address*	Error. Throws acFileClientPickFile fault.
CIFS with wrong host name	Error. Throws acFileClientPickFile fault.
CIFS with wrong share name	Error. Throws acFileClientPickFile fault.
CIFS with wrong authentication	Error. Throws acFileClientPickFile fault.
CIFS with wrong domain name	Error. Throws acFileClientPickFile fault.
File name with unsupported characters	Error. Throws acFileClientPickFile fault.

Policy: none.

1.8.5 acFileClientRemoveFile

Caption: FileClientRemoveFile.

Throws Fault=acFileClientRemoveFile.

The acFileClientRemoveFile activity is a single file activity that removes a file specified by a file path from either the local file storage or CIFS shared folders. It uses the *removeFile* operation from the *FileSystem Adapter* component (refer to chapter *1.11.6 removeFile (Delete s a single file))* of the *BIS6 Base Components* manual or chapter *8.1 removeFile (Delete s a single file)* of the *FileSystem Adapter* manual for more details).

File path	*<FilePath>*[1]	Mandatory. Editable text field. Default value 'myfile.txt'. The path and name of the file to be removed. The *File path* supports: • Relative path in relation to the *Connection* root path. In this case, the relative path in *Address* is overwritten. • System variables (page 392), e.g. '${RELATIVE_PATH}*.dta'. • Properties (page 392), e.g. '$[RELATIVE_PATH]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.

File system type		Mandatory. Non-editable selection field. Type of file system resources to use. Affects the lists of available addresses and connections below.
	Local	Default. Local file resources.
	CIFS	CIFS shared folders on file storage resources that support the CIFS protocol using share names.
Address	*<Ref>*[2]	Mandatory. Editable selection field. Master data File Client address according to the *file system type*. The relative path to the directory to be listed, in relation to the *Conne ction* root path. The *Address* supports: • System variables (page 392), e.g. '${ADDRESS}'. • Properties (page 392), e.g. '$[ADDRESS]'.
Connection	*<Ref>*[2]	Mandatory. Editable selection field. Master data File Client connection according to the *file system type*. Means the root path: The absolute path name of a directory for Local access, or the share name (or directory path starting with the share name) for CIFS access. The *Address* supports: • System variables (page 392), e.g. '${CONNECTION}'. • Properties (page 392), e.g. '$[CONNECTION]'.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid relative file path and valid file name.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- File client adapter started.

Result:

- Current process attachment: not accessed and not changed.

Sets the following properties:

- TARGET_PATH: The path of the file that has been removed.
- TARGET_NAME: The name of the file that has been removed.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acFileClientRemoveFile fault.
Directory without permission	Error. Throws acFileClientRemoveFile fault.
File not existing	Error. Throws acFileClientRemoveFile fault.
Empty file	Success. Deletes the file.
File without permission	Error. Throws acFileClientRemoveFile fault.
Locked file	Error. Throws acFileClientRemoveFile fault.
Not existing *Connection*	Error. Throws acFileClientRemoveFile fault.
Not existing *Address*	Error. Throws acFileClientRemoveFile fault.
CIFS with wrong host name	Error. Throws acFileClientRemoveFile fault.
CIFS with wrong share name	Error. Throws acFileClientRemoveFile fault.
CIFS with wrong authentication	Error. Throws acFileClientRemoveFile fault.
CIFS with wrong domain name	Error. Throws acFileClientRemoveFile fault.
File name with unsupported characters	Error. Throws acFileClientRemoveFile fault.

Policy: none.

1.8.6 acFileClientRename

Caption: FileClientRename.

Throws Fault=acFileClientRename.

The *acFileClientRename* activity is a single file activity used to rename a file. It uses the *renameFile* operation from the *FileSystem Adapter* component (refer to chapter *1.11.7 renameFile (Rename s a single file)* of the *BIS6 Base Components* manual or chapter *8.2 renameFile (R ename s a single file)* of the *FileSystem Adapter* manual for more details).

Mode of operation:
Renames the specified source file as the specified target file on a local file system or a CIFS shared folder. The target file can be located in another directory. This target directory is always a subdirectory of the same root directory as the source file. The current attachment cannot be used as source file.

Field name	Value	Description
Source file path	<FilePath>[1]	Mandatory. Editable text field. Default value 'myfile.txt'. The (relative) path and name of the source file. The *Source file path* supports: • Relative file path in relation to the *Connection* root path, overwrites the relative *Address* path. Can be omitted. • System variables (page 392), e.g. '${RELATIVE_PATH}*.dta'. • Properties (page 392), e.g. '$[MY_FILE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.

Field name	Value	Description
Target file path	<FilePath>[1]	Mandatory. Editable text field. Default value 'myfile.txt'. The (relative) path and name of the target file in relation to the *Connection* root path. The *Target file path* supports: • Relative file path in relation to the *Connection* root path, overwrites the relative *Address* path. Can be omitted. • System variables (page 392), e.g. '${RELATIVE_PATH}*.dta'. • Properties (page 392), e.g. '$[MY_FILE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
File system type		Mandatory. Non-editable selection field. Type of file system resources to use. Affects the lists of available *Addresses* and *Connections* below.
	Local	Default. Local file resources.
	CIFS	CIFS shared folders on file storage resources that support the CIFS protocol using share name.
Address	*<Ref>*[2]	Mandatory. Editable selection field. Master data File Client address according to the *File system type*. The relative path to the source and target directory in relation to the *Connection* root path. The *Address* supports: • System variables (page 392), e.g. '${ADDRESS}'. • Properties (page 392), e.g. '$[ADDRESS]'.
Connection	*<Ref>*[2]	Mandatory. Editable selection field. Master data File Client connection according to the *File system type*. The source and target root path: The absolute path name of a directory for local access, or the share name (or directory path starting with the share name) for CIFS access. The *Connection* supports: • System variables (page 392), e.g. '${CONNECTION}'. • Properties (page 392), e.g. '$[CONNECTION]'.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid relative file path and valid file name.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- File client adapter started.

Result:

- Current process attachment: not accessed and not changed.

Sets the following properties:

- TARGET_NAME := The name of the target file.
- TARGET_PATH := The path of the target file.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing source directory	Error. Throws acFileClientRename fault.
Not existing target directory	Error. Throws acFileClientRename fault.
Source directory without permission	Success. Source file is renamed.
Target directory without permission	Error. Throws acFileClientRename fault.
Empty source file	Success. Source file is renamed.
Source file not existing	Error. Throws acFileClientRename fault.
Source file without permission	Error. Throws acFileClientRename fault.
Locked source file	Error. Throws acFileClientRename fault.
Already existing target file	Error. Throws acFileClientRename fault.
Not existing *Connection*	Error. Throws acFileClientRename fault.
Not existing *Address*	Error. Throws acFileClientRename fault.
CIFS with wrong host name	Error. Throws acFileClientRename fault.
CIFS with wrong share name	Error. Throws acFileClientRename fault.
CIFS with wrong authentication	Error. Throws acFileClientRename fault.
CIFS with wrong domain name	Error. Throws acFileClientRename fault.
Source file name with unsupported characters	Error. Throws acFileClientRename fault.
Target file name with unsupported characters	Error. Throws acFileClientRename fault.

Policy: none.

1.8.7 acDeleteFile

Caption: Delete file (hotfolder).

Throws Fault=acDeleteFile

The acDeleteFile activity is a single file activity that removes a file specified by a file path from the local file storage without reference to master data. It uses the *removeFile* operation from the *StoreNode* component (refer to chapter *6.6.1 removeFile* of the *BIS Base Components* manual for more details).

Path	<FilePath>[1]	Mandatory. Editable text field. Empty ba default. The absolute path and name of the file to be removed. The *Path* supports: • System variables (page 392), e.g. '${MY_FILE_PATH}', • Properties (page 392), e.g. '$[MY_FILE_PATH]' . • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).

		Wildcards are not supported.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid absolute file path and valid file name.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- HotfolderService adapter started.

Result:

- Current process attachment: not accessed and not changed.

The following properties are set:

- TARGET_PATH: The path of the file that has been removed.
- TARGET_NAME: The name of the file that has been removed.

Possible cases of failure:

Possible case of failure	Success or error behavior
Directory not existing	Error. Throws acDeleteFile fault.
Directory without permission	Error. Throws acDeleteFile fault.
File not existing	Error. Throws acDeleteFile fault.
Empty file	Success. Deletes the file.
File without permission	Error. Throws acDeleteFile fault.
Locked file	Error. Throws acDeleteFile fault.
File name with unsupported characters	Error. Throws acDeleteFile fault.

Policy: none

1.8.8 acPickFile

Caption: Pick file (hotfolder).

Throws Fault=acPickFile.

The acPickFile activity is a single file activity that creates an attachment from the specified file located on a local folder. It uses the *getFile* operation from the *FileSystem Adapter* component (refer to chapter *1.11.2 getFile (Retrieves file as attachment)* of the *BIS6 Base Components* manual or chapter *8.3 getFile (Retrieves file as attachment)* of the *FileSystem Adapter* manual for more details).

Source path	<FilePath>[1]	Mandatory. Editable text field. Empty by default. The absolute path to the parent directory of the source file. The *Source path* supports:

		• System variables (page 392), e.g. '${MY_FILE_PATH}'. • Properties (page 392), e.g. '$[MY_FILE_PATH]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
File name	<FileName>[2]	Mandatory. Editable text field. Empty by default. The valid name of the source file. The *File name* supports: • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Delete picked file	True	Delete the picked file.
	False	Default. Keep the picked file.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[3]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Valid absolute directory path.
2) Valid file name.
3) Reference to master data
4) Arbitrary text

Preconditions:

- File client adapter started.

Result:

- New process attachment: Picked file.

Sets the following properties:

- TARGET_NAME := The name of the picked file.
- TARGET_PATH := The path of the picked file.
- FILE_COUNT := The number of files that have been picked.

Possible cases of failure:

Possible case of failure	Success or error behavior
Directory not existing	Error. Throws acPickFile fault.
Directory without permission	Error. Throws acPickFile fault.
File not existing	Error. Throws acPickFile fault.
Empty file	Success. Picks the file.
File without permission	Error. Throws acPickFile fault.
Locked file	Error. Throws acPickFile fault.
File name with unsupported characters	Error. Throws acPickFile fault.

Policy: none

1.8.9 mdFileSystem

Caption: Store file (file system).

Throws Fault=acForwardingFileSystem

The *mdFileSystem* activity is a single file activity that allows the current process attachment to be stored to either the local file storage or to CIFS shared folders. It uses the *storeAttachmentToFile* operation from the *FileSystem Adapter* component (refer to chapter *1.11.8 storeAttachmentToFile (Store s an attachment to file)* of the *BIS6 Base Components* manual) or chapter *8.5 storeAttachmentToFile (Store s an attachment to file)* of the *FileSystem Adapter* manual for more details).

Mode of operation:
The current process attachment is stored to either the local file system or to CIFS shared folders. Handling of already existing target files can be configured.

Field name	Value	Description
Path	*<FilePath>*[1]	Optional. Editable text field. Empty by default. The relative path to the parent directory of the file, in relation to the root path specified by the *File client connection*. Overwrites the *File client address* path. The *Path* supports: • System variables (page 392), e.g. '${MY_FILE_PATH}'. • Properties (page 392), e.g. '$[MY_FILE_PATH]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
File name	*<FileName>*[2]	Mandatory. Editable text field. Empty by default. The name of the target file to be stored. The *File name* supports: • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.

Field name	Value	Description
File system type		Mandatory. Non-editable selection field. Type of file system resources to use, affects the types of available *File client addresses* and *File client connections* below.
	Local	Default. Local file resources.
	CIFS	CIFS shared folders on file storage resources that support the CIFS protocol using share names.
File client address	*<Ref>*^{3}	Mandatory. Editable selection field. Master data File client address according to the file system type. Denotes the relative path to the parent directory of the file, in relation to the root path specified by the *File client connection*. The *Address* supports: • System variables (page 392), e.g. '${ADDRESS}'. • Properties (page 392), e.g. '$[ADDRESS]'.
File client connection	*<Ref>*^{3}	Mandatory. Editable selection field. Master data File client connection according to the *File system type*. Means the root path: the absolute path name of a directory or the share name (or directory starting with the share name) for CIFS access. The *Connection* supports: • System variables (page 392), e.g. '${CONNECTION}'. • Properties (page 392), e.g. '$[CONNECTION]'.
Append mode		Mandatory. Non-editable selection field. Provides additional options for the file name of the target file and determines what to do in case there is already an existing file with the same name. Assumes that the file name is specified in the process, or in the master data. In each case, the target file with the name *File name* is created if it does not yet exist.
	Append	If a file with the name *File name* already exists in the directory, the new file data will be appended to the existing file.
	Create new file	Default. If a file with the name *File name* already exists in the directory, it will be kept and the operation will finish with a fault, without creating a new file.
	Overwrite	If a file with the name *File name* already exists in the directory, it will be overwritten.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*^{3}	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*^{4}	Optional. Refer to Common Fields (page 3).

1) Valid relative directory path.
2) Valid file name.
3) Reference to master data.
4) Arbitrary text.

Preconditions:

- File client adapter started.
- Available process attachment.

Result:

- Current process attachment: not changed.

The following properties are set in cases of success:

- TARGET_NAME := The name of the target file.
- TARGET_PATH := The path of the target file.
- TO_ADAPTER := 'fileClient.storeFile'.
- TO_NAME := sToName ##.
- DIRECTION := 'OUT'.

Possible cases of failure (ACO denotes the relevant *Append mode*s - A=Append, C=Create new file, O=Overwrite):

Possible case of failure	**Append mode**	**Success or error behavior**
Not existing directory	ACO	Success. Each missing directory is created and file is stored.
Directory without permission	ACO	Error. Throws acForwardingFileSystem fault.
Already existing file	AO	Success. Existing file is being appended (A) or overwritten (O).
Already existing file	C	Error. Throws acForwardingFileSystem fault.
File without permission	AO	Error. Throws acForwardingFileSystem fault.
File without permission	C	Not relevant, since files without permissions are treated as readable. The case actually matching is 'Already existing file'.
Locked file	AO	Error. Throws acForwardingFileSystem fault.
Locked file	C	Not relevant, since locked files are treated as readable. The case actually matching is 'Already existing file'.
Not existing *Connection*	ACO	Error. Throws acForwardingFileSystem fault.
Not existing *Address*	ACO	Error. Throws acForwardingFileSystem fault.
CIFS with wrong host name	ACO	Error. Throws acForwardingFileSystem fault.
CIFS with wrong share name	ACO	Error. Throws acForwardingFileSystem fault.
CIFS with wrong authentication	ACO	Error. Throws acForwardingFileSystem fault.
CIFS with wrong domain name	ACO	Error. Throws acForwardingFileSystem fault.
File name with unsupported characters	ACO	Error. Throws acForwardingFileSystem fault

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT-FILESYSTEM ALLOWANCE/TRANSPORT/FILESYSTEM

1.8.10 mdStoreFile

Caption: Store file (hotfolder).

Throws Fault=acForwardingHotfolder

The *acForwardingHotfolder* activity is a single file activity that allows the current attachment to be stored to the local file system without using any master data configuration. It uses the *storeAttachments* operation from the *StoreService* component (refer to chapter *6.7.4 storeAttachments* of the *BIS6 Base Components* manual) for more details).

Field name	Value	Description
Path	*<FilePath>*[1]	Mandatory. Editable text field. Empty by default. Provides a directory selection dialog. The absolute path to the parent directory of the target file. The *Path* supports: • System variables (page 392), e.g. '${MY_PATH}*.dta'. • Properties (page 392), e.g. '$[RELATIVE_PATH]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
File name	*<FileName>*[2]	Mandatory. Editable text field. Empty by default. The file name for the target file to be stored. Must be a valid file name. The *File name* supports: • System variables (page 392) e.g. '${MY_FILE_NAME} • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Append mode		Mandatory. Non-editable selection field. Property with four different modes that shows how to handle the case when a file has already been created in the target location. In each case, the target file is created if it does not yet exist.
	Autorename	If a file with the name *File name* already exists, a new file is created whose name is combined from the target *File name* and a 13-digits extension denoting the file timestamp in milliseconds, for instance `hottestAutorename_1447237302435 .dta` .
	Compare	Default. If a file with the name *File name* already exists, this existing file and the new file are compared: • Identical file content: This activity finishes with success. A different timestamp is ignored. • Different file content: This activity stops with error, throws an *acForwardingHotfolder* fault.
	Fail	If a file with the name *File name* already exists, the activity stops with error and throws an *acForwardingHotfolder* fault.
	Overwrite	If a file with the name *File name* already exists, it is overwritten.

Field name	Value	Description
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[3]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Valid absolute directory path.
2) Valid file name.
3) Reference to master data
4) Arbitrary text.

Preconditions:

- StoreService adapter started.
- Available process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- TARGET_NAME := The name of the target file.
- TARGET_PATH := The path of the target file.

Possible cases of failure (ACFO denotes the relevant *Append mode*s - A=Autorename, C=Compare, F=Fail, O=Overwrite):

Possible case of failure	Append mode	Success or error behavior
Not existing directory	ACFO	Success. Each missing directory is created.
Directory without permission	ACFO	Error. Throws acForwardingFileSystem fault.
Already existing file	AO	Success. Existing file is being appended (A) or overwritten (O).
	C	Result depends on file content: • Identical: Success. Nothing done since not necessary. • Different: Error. Throws acForwardingFileSystem fault.
	F	Error. Throws acForwardingFileSystem fault.
File without permission	AFO	Error. Throws acForwardingFileSystem fault.
	C	Result depends on file content: • Identical: Success. Nothing done since not necessary. • Different: Error. Throws acForwardingFileSystem fault.
Locked file	AFO	Error. Throws acForwardingFileSystem fault.

Possible case of failure	Append mode	Success or error behavior
	C	Result depends on file content: • Identical: Success. Nothing done since not necessary. • Different: Error. Throws acForwardingFileSystem fault.
File name with unsupported characters	ACFO	Error. Throws acForwardingFileSystem fault

Policy:

> REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT-FILESYSTEM ALLOWANCE/TRANSPORT/FILESYSTEM

1.9 Split Activities

This section describes file splitting activities.

1.9.1 acSplitFile

Caption: SplitFile.

Throws Fault=acSplitFile.

The *acSplitFile* activity is a single file activity that splits the current attachment into several files that trigger new *MFT-SplitterEvents* (refer to chapter MFT_SplitterEvent of the *MFT Standard Solution - Adapters* manual for more details). Uses the *Split* operation from the *SplitterPT* adapter *(*refer to chapter *4.7.1 Split* of the *BIS6 Base Components* manual or the *BIC File Splitter* manual) for more details), or the *convert* operation from the *BicNode* component (refer to chapter *4.1.1 convert* of the *BIS6 Base Components* manual for more details), depending on the selected *Split type*.

Operation mode:
This activity splits the current process attachment in several parts. Each of this part triggers an MFT_SplitterEvent that starts a new process. Each event gets all properties from this activity. The FROM_ADAPTER property is set to 'SPLFIL' which corresponds to the 'File splitter' adapter available in the *From adapter* selection field of the MFT-Criteria.

Field name	Value	Description
Split type		Mandatory. Non-editable selection field. Provides supported split modes.
	Use mapping	Use a BIC mapping for handling the attachment. Displays the *Mapping* field for mapping master data selection.
	Use splitter	Default. Use a splitter configuration for handling the attachment. Displays the *Configuration* field for splitter master data selection.
Configuration	*<Ref>*[1]	Mandatory. Non-editable selection field. Master data splitter configuration. The splitter to use for splitting the process attachment. Visible only when *Split type* is set to *Use splitter*.
Mapping	*<Ref>*[1]	Mandatory. Editable selection field. Master data mapping. The BIC mapping to use for splitting the process attachment. The *Mapping* supports:

Field name	Value	Description
		• System variables (page 392), e.g. '${MAPPING_NAME}'. • Properties (page 392), e.g. '$[MAPPING_NAME]'. Values from the split attachment parts can be added as properties to the property list of each event. Thus, it is possible to overwrite the properties used by MFT-Criteria entity lookup by setting mapping output values. Visible only when *Split type* is set to *Use mapping*.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Split event		Mandatory. Provides supported modes for triggers a new MFT_SplitterEvent for each splitter/mapping output.
	Initiate new process	Default. Sole selection entry by now. Creates a new process that is started with the MFT_SplitterEvent.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.

Preconditions:

- BIC Splitter Node adapter started if *Split type* is set to 'Use splitter' .
- BIC Client adapter started if *Split type* is set to 'Use mapping'.
- Script Executor adapter started if *Split type* is set to 'Use mapping'.
- 'GetSplitProperties.bsh' script available if *Split type* is set to 'Use mapping'.
- Available process attachment: format matching selected splitter/mapping.
- Entity catching the created MFT_SplitterEvents for the

Result:

- New MFT_SplitterEvents for the splitter/mapping outputs with all properties from this activity and FROM_ADAPTER property set to 'SPLFIL'.
  If a mapping is used values from the split attachment parts can be added as properties to the property list of each event. Thus, it is possible to overwrite the properties used by MFT-Criteria entity lookup.
- Current process attachment: unchanged.

Sets the following properties:

- SPLIT_TYPE := 'SPLITTER' if *Split type* is set to 'Use splitter',
  'MAPPING' if *Split type* is set to 'Use mapping'.
- SPLIT_EVENT := 'NEWPROCESS'.
- SPLIT_FILE_COUNT := The number of the splitter/mapping outputs.

Possible cases of failure:

Possible case of failure	Success or error behavior
BIC Splitter Node adapter not started for *Split type* 'Use splitter'	Error. Throws acSplitFile fault.
BIC Client adapter not started for *Split type* 'Use mapping'	Error. Throws acSplitFile fault.
Script Exceutor adapter not started for *Split type* 'Use mapping'	Error. Throws acSplitFile fault.
Splitter error	Error. Throws acSplitFile fault.
Mapping error	Error. Throws acSplitFile fault.
Invalid property in *Mapping*	Error. Throws acSplitFile fault.
Invalid system variable in *Mapping*	Error. Throws acSplitFile fault.
Non-existing mapping in *Mapping*	Error. Throws acSplitFile fault.

Policy: none.

Example: *Splitting a flat file* (page 416).

1.10 Packer/Unpacker Activities

This section describes archiving, compress and uncompress related activities.

1.10.1 acCompress2

Caption: Compress.

Throws Fault=acCompress2.

The *acCompress2* activity is a file activity that offers several archiving and compression modes.

Field name	Value	Description
Method		Mandatory. Non editable selection field. Provides several archiving and compression modes.
	GZIP	Default. gzip compression with gzip file format. Offers single file compression.
	PKZIP	PKZIP archiving with ZIP file format. Offers multiple file archiving and compression.
	External compress program	Calling external archiver.
	Pack local folder	Archiving with ZIP file format. Offers directory and directory tree archiving and compression as well as password protection.

Compress method: GZIP.

This compress mode is a single file mode that compresses the current process attachment. Uses the *compre ss* operation of the *Compression* component (refer to chapter *6.1.1 compress* of the *BIS6 Base Components* manual for more details).

Field name	Value	Description
File name	*<FileName>*[1]	Optional. Empty by default. Editable text field. File name to use for the current process attachment inside the resulting gzip archive. The *File name* supports: • Properties (page 392), e.g. '$[COMPRESS_FILE_NAME]'. • System variables (page 392), e.g. '${COMPRESS_FILE_NAME}'. • XPath expressions (page 392), limited support due to invalid characters, e.g. $(upper-case('MyFileName.txt')). Invalid characters \ / & : ; ?
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Arbitrary text.

Preconditions:

- Available process attachment.

Result:

- New process attachment: compressed file in gzip file format.

Sets the following properties:

- RESULT := Used method from *Method* field, i.e. 'GZIP'.
- COMPRESS_METHOD := Used method from *Method* field, i.e. 'GZIP'.
- COMPRESS_SIZE := Size of compressed file.

Compress method: PKZIP.

This compress mode is a multiple file mode that archives or compresses the current process attachment and a list of bookmarked attachments. Uses the *compressArchive* operation of the *Compression* component (refer to chapter *6.1.2 compressArchive* of the *BIS6 Base Components* manual for more details).

Field name	Value	Description
File path in zip file	*<FilePath>*[1]	Mandatory. Editable text field. Default value '$[FROM_PATH]. File name to use for the current process attachment in the zip file. Can include a file path. The *File path in zip file* supports: • Properties (page 392), e.g. '$[ZIP_FILE_PATH]'. • System variables (page 392), e.g. '${ZIP_FILE_PATH}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Bookmarks to add (filepath=<bookmark>;...)	*<KVList>*[2]	Optional. Editable text field. Empty by default. List of bookmarked attachments that are added to the zip file with the file path the bookmark is assigned to. The *Bookmarks to add zip file* supports:

Field name	Value	Description
		• Properties (page 392), e.g. '$[FILE_PATH]=BOOKMARK;'. • System variables (page 392), e.g. '${FILE_PATH}=BOOKMARK;' • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). • Maximum number of key-value pairs is 100. Additional key-value pairs are ignored. ⚠ The file paths in this bookmark list need to be different. Identical file paths are overwritten.
Compression level		Mandatory. Non editable selection field. Provides several archiving and compression modes.
	No compression	File archiving without compression.
	<1-5>	File archiving with low compression level.
	6	Default compression level.
	<7-9>	File archiving with high compression level.
Password from property	True	Makes *Encryption password clear text* visible.
	False	Default. Makes *Encryption password* visible.
Encryption password clear text	*<Password>*[3]	Mandatory. Editable text field that displays the password in plain text. Empty by default. Visible when *Password from property* is switched on.
Encryption password	*<Password>*[3]	Optional. Editable password field. Empty by default. Properties, system variables and Xpath expressions are not supported. Visible when *Password from property* is switched off.
Encryption method		Mandatory. Non editable selection field. Provides supported encryption methods for *Encryption password* or *Encryption password clear text*.
	STANDARD	Default. Standard zip encryption.
	AES128	Advanced Encryption Standard. Block cipher with 128 bits block size and 128 bits key length.
	AES256	Advanced Encryption Standard. Block cipher with 128 bits block size and 256 bits key length.
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Valid file path and valid file name. File path is optional. File name is mandatory.
2) Semicolon separated key-value list assigning existing bookmarks to file paths.
3) Secure password.
4) Arbitrary text.

Preconditions:

- Available process attachment.

Result:

- New process attachment: compressed file in zip file format.

Sets the following properties:

- RESULT := Used method from *Method* field, i.e. 'PKZIP'.
- COMPRESS_METHOD := Used method from *Method* field, i.e. 'PKZIP'.
- COMPRESS_SIZE := Size of archived/compressed file.

Compress method: External compress program.

This compress mode is a single file mode that compresses the current process attachment. Calls an external program whose access path is stored in the system variable '${MFT-PKZIP-COMPRESS-PROGRAM}'.

Field name	Value	Description
Zip timeout	*<Int>*[1]	Mandatory. Editable number field. Default value 60 seconds. Timeout in seconds.
Command line switches		Mandatory. Editable text field. Default value '-passphrase= %PASSWORD% %ZIP_FILE% %SOURCE_FILE%'. Command line for the external program whose access path is stored in the system variable '${MFT-PKZIP-COMPRESS-PROGRAM}'. The default value contains placeholders: • %PASSWORD%: replaced by value from *Password*; removed if Password is empty. The password for decrypting the zipped file if desired. • %ZIP_FILE%: replaced by a temporary folder in the current working directory stored in the '${MFT-ZIP-WORKDIR}' system variable. The path contains the TRACKID '<WorkDir>/<TrackID>', the file name is '<TrackID>.zip'. The folder and the file are deleted after *acCompress2* has finished. • %SOURCE_FILE%: replaced by a temporary folder in the current working directory, the path contains the TRACKID '<WorkDir>/<TrackID>', the file name is the original file name stored in the '$[FROM_PATH]' property. The referenced file is the current process attachment. Refer to the manual of your archiver for more details about the command line syntax. The *Command line switches* supports: • Properties (page 392), e.g. '$[COMPRESS_FILE_NAME]'. • System variables (page 392), e.g. '${COMPRESS_FILE_NAME}'. • XPath expressions (page 392), limited support due to invalid characters, e.g. $(upper-case('MyFileName.txt')).
Password from property	True	Makes *Encryption password clear text* visible.
	False	Default. Makes *Encryption password* visible.
Encryption password clear text	*<Password>*[2]	Mandatory. Editable text field that displays the password in plain text. Empty by default. Visible when *Password from property* is switched on.

Field name	Value	Description
Encryption password	*<Password>*[2]	Optional. Editable password field. Empty by default. Properties, system variables and XPath expressions are not supported. Visible when *Password from property* is switched off.
Debug level		Mandatory. Non editable selection field. Provides 3 debug levels.
	<default>	Default level.
	No debug	Default. Standard level. No logging of script call. No console output and no logging attachment.
	Debug	Simple debug level. Script call is logged. No console output. Logging attachment is created. Log saved to disc to main mft.log file as configured by system variable (page 392) ${MFT-LOG-SUB-PATH}.
	Verbose	Elaborate debug level. Script call is logged. Console output is written. Logging attachment is created. Log saved to disc to main mft.log file as configured by system variable (page 392) ${MFT-LOG-SUB-PATH}.
Queue	*<Ref>*[3]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Integer number.
2) Secure password.
3) Reference to master data.
4) Arbitrary text.

Additional configuration:

System variable	Description
MFT-ZIP-WORKDIR	Valid working directory.
MFT-PKZIP-COMPRESS-PROGRAM	Access path to installed external compression program.

Preconditions:

- Available process attachment if '%SOURCE_FILE%' placeholder is used in *Command line switches*.
- 'mftCallExternalZip.bsh' script existing.
- Installed compression program from '${MFT-PKZIP-COMPRESS-PROGRAM}' system variable with command line interface.

Result:

- New process attachment: compressed file in zip file format.

Sets the following properties:

- RESULT := Used method from *Method* field, i.e. 'EXTERNAL'.
- COMPRESS_METHOD := Used method from *Method* field, i.e. 'EXTERNAL'.
- COMPRESS_SIZE := Size of archived/compressed file.

Compress method: Pack local folder.

This compress mode is a multiple file mode that archives or compresses a directory or a directory tree and offers additional password protection.

Mode of operation:
First, the *Include pattern* is evaluated for all files of the specified directory or directory tree. Pattern evaluation takes place on the actual filename, the file path is not included. If the optional *Exclude pattern* is filled, it is evaluated for all files from the result set of the first step. The final result set contains all files whose names do not match the *Exclude pattern*.

Field name	Value	Description
Folder to zip	*<DirPath>*[1]	Mandatory. Default value empty. Editable text field. The path of the source folder that is to be compressed. The *Folder to zip* supports: • Properties (page 392), e.g. '$[DIR_TO_ZIP]'. • System variables (page 392), e.g. '${DIR_TO_ZIP}'.
Recursive		Mandatory. Non editable selection field.
	No	Default. No recursion. Processes only files on the source folder level.
	Recursive, flatten path in zip	Processes complete directory tree. Use files without path.
	Recursive with sub-path in zip	Processes complete directory tree. Keep tree structure in zip file.
Delete files after success		Mandatory. Non editable selection field. Provides several modes for keeping or deleting the compressed files. Deletion takes place after successfully creating the resulting compressed attachment.
	Delete files	Default. Delete compressed files.
	Delete zip folder (recursive)	Recursively delete the compressed folders.
	Keep files	Keep all created files.
Exclude pattern	*<RegExpr>*[2]	Optional. Editable text field. Default value empty. File list filter acting as exclude pattern. Processes only files that do not match this condition.The *Exclude pattern* is evaluated after the *Include pattern* if existing. The *Exclude pattern* supports: • Properties (page 392), e.g. '$[MY_PATTERN]'. • System variables (page 392), e.g. '${MY_PATTERN}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')) The kind of pattern depends on the currently used pattern mode. Refer to **File and Directory Tree Patterns** section below for more details about available pattern modes and their usage.
Include pattern	*<RegExpr>*[2]	Mandatory. Editable text field. Default value '*.*'. File list filter acting as include pattern. Processes only files that match this condition. The *Include pattern* is evaluated at first. The *Exclude pattern* is evaluated afterwards if existing. The *Include pattern* supports: • Properties (page 392), e.g. '$[MY_PATTERN]'. • System variables (page 392), e.g. '${MY_PATTERN}'.

Field name	Value	Description
		• XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')) 2 pattern modes are available. Refer to **File and Directory Tree Patterns** section below for more details about available pattern modes and their usage.
Password from property	True	Makes *Encryption password clear text* visible.
	False	Default. Makes *Encryption password* visible.
Encryption password clear text	*<Password>*[3]	Mandatory. Editable text field that displays the password in plain text. Empty by default. Visible when *Password from property* is switched on.
Encryption password	*<Password>*[3]	Optional. Editable text field. Password for creating a password-protected zip file. File extension is '.enc'. Properties, system variables and XPath expressions are not supported. Visible when *Password from property* is switched off.
Queue	*<Ref>*[4]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[5]	Optional. Refer to Common Fields (page 3).

1) Valid directory path.
2) Valid regular expression.
3) Secure password.
4) Reference to master data record.
5) Arbitrary text.

Additional configuration:

System variable	Description
MFT-RUN-COMPRESS-FOLDER-SCRIPT	Name of the BSH script to be executed.

Preconditions:

- Script Executor adapter started.
- The BSH script from the ${MFT-RUN-COMPRESS-FOLDER-SCRIPT} system variable available in the ressource repository in '<LS>/scripts/bsh/1' (The ressource repository search uses the current logical system, the BEANSHELL type, and the name '1/<*Script name*>', where '1' is the fix version number).

Result:

- New process attachment: compressed file in zip file format.
  File extension '.zip' if not password protected.
  File extension '.enc' otherwise.

Sets the following properties:

- RESULT := Used method from *Method* field, i.e. 'ZIPFOLDER'.
- COMPRESS_METHOD := Used method from *Method* field, i.e. 'ZIPFOLDER'.
- COMPRESS_SIZE := Size of compressed file.

Possible cases of failure:

Possible case of failure	Compression mode	Success or error behavior
Empty file	GZIP	Success. File is archived.
Invalid file path for *File path in zip file*	PKZIP	
Non existing bookmark in *Bookmarks to add*	PKZIP	Error. Throws acCompress2 fault.
Invalid format in *Bookmarks to add*	PKZIP	Error. Throws acCompress2 fault.
Identical file paths in *Bookmarks to add*	PKZIP	Success. The last bookmark assignment wins and overwrites its predecessors.
File path in *Bookmarks to add* is identical to *File path in zip file*	PKZIP	Error. Throws acCompress2 fault.
Invalid command line	EXTERNAl	Error. Throws acCompress2 fault.
Invalid file path for *Folder to zip*	ZIPFOLDER	Error. Throws acCompress2 fault.
Non existing file path for *Folder to zip*	ZIPFOLDER	Error. Throws acCompress2 fault.
Folder to zip without access permissions	ZIPFOLDER	Error. Throws acCompress2 fault.
Empty *Folder to zip*	ZIPFOLDER	Success. Creates empty zip file.
Non-empty *Folder to zip* with at least 1 file without access permissions	ZIPFOLDER	Error. Throws acCompress2 fault.

Policy: none.

File and Directory Tree Patterns

2 different modes are available providing support for regular expression (regex) and for glob syntax. The default mode is the regex mode. The mode can be switched by editing the system variable (page 392) ${MFT-RUN-COMPRESS-FOLDER-SCRIPT} that holds the BSH script used for finding the matching files.

- The default value is 'ZipFileList.bsh' that supports regex syntax.
- The second script is 'ZipFileList2.bsh' that supports glob syntax.

Regex mode

This section gives a short overview of regular expressions and how they are used by this activity.
Regular expressions used for file list filtering in this activity are slightly different from standard regular expressions. Some basics:

- Each pattern is implicitly preceded by '\s' in the script which represents a white space. Thus, always start the regex pattern with a '*' or '?' wildcard if a blank in the name is not desired. Otherwise, only files with a blank in the name are matching.
- A regex pattern matches anywhere in the filename unless anchored to the beginning (not supported) or to the end (supported).
- Supports the following metacharacters: [ ] ( ) { } | . * ? + - \ ^ $
- All other characters are literal characters
- Metacharacters can be escaped with a backslash in order to interprete them literally.

- Standard regex patterns are case sensitive. Case insensitivity can be switched on by preceding the pattern with '(?i)'.

Metacharacters

Character	Name	Type	Description
.	Dot	Single character match	Matches exactly 1 character, any character except newline. Interpre literally within the square bracke [...]
?	Question mark	Quantification	Preceding pattern element is optional, i.e. occurs 0 or 1 times.
*	Asterisk	Quantification	Preceding pattern element occurs 0 or more times.
+	Plus sign	Quantification	Preceding pattern element occurs 1 or

Character	Name	Type	Description
			more times.
^	Caret	Anchor	The caret '^' representing the anchor for the beginning of the filename.
$	Dollar sign	Anchor	Matches the end of file name.
[...]	Bracket expression	Single character selection	Matches exactly 1 character from inside the brackets.
	Character set		Matches 1 character from the enumerated set, e.g. [abc] matches 'a' or 'b' or 'c'
	Character range		Matches 1 character from the specified

Character	Name	Type	Description
			range, e.g. [-z] matches any lower case letter from 'a' to 'z'
	Mixed bracket		Set and range can be mixed, e.g. [0-9a-z] matches any number or any lower case letter from 'a' to 'z'
[^...]	Caret	Negation	Matches exactly 1 character except those characters from inside the brackets
|	Pipe	Choice	Separates 2 or more alternatives

Character	Name	Type	Description
\	Backslash	Escape character	Masks succeeding metacharact in order to use them literally
(...)	Parentheses	Sub expression	Groups sub patterns, e.g. Nov(ember) matches 'Nov' and 'November'
{min,max}	Curly braces	Quantification	Specifies specific amount of repetition. Preceding pattern element occurs at least *min* and not more than *max* times

Basic Include patterns

Pattern	Match Description	No match
?	All files	
*	All files	

Pattern	Match Description	No match
+	All files containing a blank ' ' in the name	Files without a blank in the name
ftpScript\.dta	All files containing ' ftpScript.dta' in the name, e.g. 'another ftpScript.dta' or	'ftpScript.dta' or 'sftpScript.dta' or 'ftpScript.dta.org'
?^ftpScript\.dta$	Exactly matches file ‘ftpScript.dta’ and nothing else	

Frequent use cases for the Include pattern
Note that the preceding wildcard which is necessary is omitted in the example patterns.

Pattern	Match Description
.*\.*	All files with a dot in the file name. Since the dot is a metacharacter it needs to be escaped if it is a literal dot.
.*	All files with any characters, minimum length of file name 1.
ftpScript\.dta	All files whose names contain 'ftpScript.dta', i.e. matches file such as 'ftpScript.dta', 'sftpScript' and 'anotherftpScript.dta' as well as 'ftpScript.dtax' or 'ftpScript.dta.old'
ftpScript.dta	All files whose names contain something like 'ftpScript.dta' or 'ftpScript_dta' or 'ftpScript1dta'.
ftpScript\.dta$	All files whose names end with 'ftpScript.dta'.
ftpScript..\.dta	All files whose names contain something like 'ftpScript01.dta' or 'ftpScript02.dta' or 'ftpScriptXY.dta'. The first 2 dots are metacharacters standing for any single character. The last dots stands for the literal dot character.
ftpScript \(2\)\.dta	All files whose names contain 'ftpScript (2).dta'. The 3 metacharacters ( ) . need to be escaped in order to be literal.
ftpScript0[^2]\.dta	All files whose names do not contain 'ftpScript02.dta' but something like 'ftpScript01.dta' or 'ftpScript0x.dta'. The bracket expression matches a single character, namely all characters except '2'.
ftpScript??\.dta	All files whose names contain something like 'ftpScript.dta' or 'ftpScript1.dta' or 'ftpScriptx.dta' or 'ftpScript12.dta' or 'ftpScriptxy.dta'. The 2 metacharacters '??' stand for any character sequence from length 0 to 2.
ftpScript0[0-9]\.dta	All files whose names contain 'ftpScript00.dta' or 'ftpScript01 .dta' or ... 'ftpScript09.dta'.
ftpScript0[012]\.dta	All files whose names contain 'ftpScript00.dta' or 'ftpScript01.dta' or 'ftpScript02.dta'.
.*\.dta	All files with extension '.dta'. By default, pattern evaluation is case sensitive.
.*\.DTA	All files with extension '.DTA'. By default, pattern evaluation is case sensitive.
(?i).*\.dta	All files with extensions '.dta' or '.dtax' or '.DTA' or '.DTAx' etc. The subexpression '(?i)' switches from case sensitivity to case insensitivity.
.*\.(?i)dta$	All files whose names end with extensions '.dta' or '.DTA' or '.Dta' etc..
*\.(htm|html|xhtml)	All files with extensions '.htm' or '.html' or '.xhtml'.

Glob mode

A glob pattern specifies a set of filenames using wildcard characters. The following character sequences have a special meaning within a glob pattern:

Character sequence	Name	Description
?	Wildcard	Matches exactly 1 character. Wildcard characters '*' - representing zero or more symbols, and '?' - representing exactly one symbol. Interpre literally within the square bracket [...]
*	Wildcard	Matches any number of character. Wildcard characters '*' - representing zero or more symbols, and '?' - representing exactly one symbol. Interpre literally within the

Frequent use cases:

Glob pattern	Matching files
*	All files
.	All files with a dot '.' in the name
*.html	All files with extension '.html'
*.{htm,html,xhtml}	All files with extensions '.htm' or '.html' or '.xhtml'
*.{txt,log}	
_*	All files whose names start with an underscore '_'
[^#]*	All files whose names do no start with a hash mark '#'

1.10.2 acUncompress

Caption: Uncompress.

Throws Fault=acUncompress.

The *acUncompress* activity is a multiple mode activity that offers several unpacking and uncompression modes.

Field name	Value	Description
Method		Mandatory. Non editable selection field. Provides several archiving and compression modes.
	UNGZIP	Default. gzip uncompression. Unpacks exactly 1 file from an gzip archive to the process attachment.
	UNZIP (single file - in process)	Zip uncompression. Unpacks exactly 1 file from a zip archive to the process attachment, configurable via file name pattern.
	UNZIP (multiple files - to folder)	Zip uncompression. Unpacks all files from a zip archive to a target folder. Offers decryption of password protected archives.
	PKUNZIP (single file - in process)	PKZIP uncompression. Unpacks exactly 1 file from a zip archive to the process attachment, configurable via file name pattern. Offers decryption of password protected archives.

Uncompress method: UNGZIP.

This compress mode is a single file mode that unpacks a single file from the current process attachment (gzip archive). Uses the de*compress* operation of the *Compression* component (refer to chapter *6.1.3 decompress* of the *BIS6 Base Components* manual for more details).

Field name	Value	Description
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.

Preconditions:

- Available process attachment: gzip format.

Result:

- New process attachment: uncompressed file.

Sets the following properties:

- FILE_COUNT := 1. Number of returned attachments.
- FILES_LEFT := remaining number of matching files.
- RESULT := 'UNGZIP'.
- UNCOMPRESS_COUNT := 1. Number of unpacked files.
- UNCOMPRESS_FILESIZE := size of unpacked file.
- UNCOMPRESS_METHOD := 'UNGZIP'. Uncompression method.

Possible cases of failure:

Possible case of failure	Success or error behavior
Process attachment without gzip format	Error. Throws acUncompress fault.

Policy: none.

Uncompress method: UNZIP (single file - in process).

This compress mode is a single file mode that unpacks a single and configurable file from the current process attachment (zip archive). Uses the de*compressArchive* operation of the *Compression* component (refer to chapter *6.1.4 decompressArchive* of the *BIS6 Base Components* manual for more details).

Field name	Value	Description
Zip entry	*<RegEx>*[1]	Optional. Empty by default. Editable text field. File name pattern matching the zip entry to be uncompressed. The empty value corresponds to the pattern '*'. Should match exactly 1 file. If more than 1 files are matching the first one in alphabetical order is used. The *Zip entry* supports: • Wildcard characters '*' - representing zero or more symbols, and '?' - representing exactly one symbol. • Character sets matching exactly one character of the specified set, e.g. [abc]. • Character ranges matching exactly one character of the specified range, e.g. [a-z]. • System variables (page 392), e.g. '${ZIP_ENTRY_NAME}'. • Properties (page 392), e.g. '$[ZIP_ENTRY_NAME]'.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Valid regular expression.
2) Arbitrary text.

Preconditions:

- Available process attachment: zip format.

Result:

- New process attachment: uncompressed file.

Sets the following properties:

- FILE_COUNT := 1. Number of returned attachments.
- FILES_LEFT := remaining number of files in archive if total count > 1.
  := 0, if the archive contains exactly 1 file.
- RESULT := 'UNZIP'.
- TARGET_NAME := filename of unpacked file without path.
  := 'NOT_SET' if the uncompressed file does not have a name.
- UNCOMPRESS_COUNT := 1. Number of unpacked files.
- UNCOMPRESS_FILESIZE := size of uncompressed file.
- UNCOMPRESS_METHOD := 'UNZIP'. Method of uncompression.

Possible cases of failure:

Possible case of failure	Success or error behavior
Process attachment without zip format	Error. Throws acUncompress fault.
Empty zip archive	Error. Throws acUncompress fault.
Empty zip entry	Success. Zip entry is unpacked.
Zip entry with blank in name	Error. Throws acUncompress fault.
Non matching zip entries	Error. Throws acUncompress fault.
More than 1 file matches	Success. The 'first' file is unpacked and returned as process attachment.

Policy: none.

Uncompress method: UNZIP (multiple files - to folder).

This uncompress mode is a multiple file mode that unpacks all files from the current process attachment (zip archive) to a target directory. Offers decryption for password protected zip archives.

Field name	Value	Description
Password	<Password>	Optional. Editable text field. Password for decrypting the current process attachment in case the zip archive is password protected.
Target folder	*<FilePath>*[1]	Mandatory. Editable text box offering directory selection dialog. Target directory where the unpacked files are to be stored.
Folder mode		Mandatory. Non editable selection field. Modes for handling zip entry file paths if existing.
	Flatten	Default. Unpack all files directly into the target directory.
	Make directories	Unpack all files into the target directory or a sub-directory keeping their file paths.
File mode		Mandatory. Non editable selection field. Provides options for the file name of the target files and determines what to do in case there is already an existing file with the same name. In each case, the target file is created if it does not yet exist.
	Overwrite	Default. If a file with the same name already exists in the target directory, it will be overwritten.
	Auto-rename	If a file with the same name already exists in the target directory, a new file is created whose name is combined from the target file name, the TRACKID (a 12-digits number) and a hyphen followed by the number of the zip entry, e.g. 'ftpScript.txt.000000000210-0'.
Queue name	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).

Field name	Value	Description
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid file path.
2) Reference to master data record.
3) Arbitrary text.

Preconditions:

- Script executor adapter started.
- 'mftUnzipAttachment.bsh' BSH script installed.
- Available process attachment: zip file format.

Result:

- Process attachment: not changed.

Sets the following properties:

- FILE_COUNT := the number of unpacked zip entries.
- FILES_LEFT := 0, in case of success.
- RESULT := 'UNZIP_MULTI'.
- UNCOMPRESS_COUNT := the number of unpacked zip entries.
- UNCOMPRESS_FILESIZE := size of unpacked file.
- UNCOMPRESS_METHOD := 'UNZIP_MULTI'. Used method for uncompression.
- TARGET_NAME := 'NOT_SET'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Process attachment without zip format	Error. Throws acUncompress fault.
Empty zip archive	Success. FILE_COUNT := 0. FILES_LEFT := 0. TARGET_NAME := 'NOT_SET'. UNCOMPRESS_COUNT := 0.
Non existing target folder	Success. Missing directory is created.
Target folder without access permissions	Error. Throws acUncompress fault.

Policy: none.

Uncompress method: PKUNZIP (single file - in process).

This uncompress mode is a multiple file mode that unpacks all files from the current process attachment (zip archive). Offers additional options such as handling password-protected zip archives. Calls an external program stored in the system variable '${MFT-PKZIP-EXTRACT-PROGRAM}'.

Field name	Value	Description
Zip entry	*<FilePattern>*[1]	Optional. Empty by default. Editable text field. File name pattern matching the zip entry to be uncompressed. The empty value corresponds to the pattern '*'. If more than 1 files are matching the first one is used. The *Zip entry* supports:

Field name	Value	Description
		• Wildcard characters '*' - representing zero or more symbols, and '?' - representing exactly one symbol. • Subfolder, e.g. 'test/test.dta'. • System variables (page 392), e.g. '${ZIP_ENTRY_NAME}'. • Properties (page 392), e.g. '$[ZIP_ENTRY_NAME]'. • Character sets matching exactly one character of the specified set, e.g. [abc]. • Character ranges matching exactly one character of the specified range, e.g. [a-z].
Password	*<Password>*[2]	Optional. Empty by default. Editable text field. The password for decrypting the zipped file. Properties, system variables and XPath expressions are not supported.
Command line switches		Mandatory. Default value '-password=%PASSWORD% %SOURCE_FILE% %ZIP_ENTRY% "%TARGET_FOLDER%"'. Editable text field. Command line options for the external program stored in the system variable '${MFT-PKZIP-EXTRACT-PROGRAM}'. The default value means if unchanged: • %PASSWORD%: replaced by *Password* value; removed if Password is empty. The password for decrypting the zipped file if necessary. • %SOURCE_FILE%: replaced by a temporary folder in the current working directory, the path contains the TRACKID '<WorkDir>\<TrackID>', the file name is '<TrackID>.zip'. • %ZIP_ENTRY%: replaced by *Zip entry* value, replaced by '*' if *Zip entry* is empty. The name or pattern of the zip entry to be extracted. • %TARGET_FOLDER%: replaced by a temporary folder in the current working directory '<WorkDir>\<TrackID>'; the working directory is stored in the system variable '${MFT-UNZIP-WORKDIR}'.
Unzip timeout	*<Int>*[3]	Mandatory. Default value 60 seconds. Editable number field. Unzip timeout in seconds.
Debug level		Mandatory. Non editable selection field. Provides 3 debug levels.
	No debug	Default. Standard level. No logging of script call. No console output and no logging attachment.
	Debug	Simple debug level. Script call is logged. No console output. Logging attachment is created. Log saved to disc to main mft.log file as configured by system variable (page 392) ${MFT-LOG-SUB-PATH}.
	Verbose	Elaborate debug level. Script call is logged. Console output is written. Logging attachment is created. Log saved to disc to main mft.log file as configured by system variable (page 392) ${MFT-LOG-SUB-PATH}.
Queue name	*<Ref>*[4]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[5]	Optional. Refer to Common Fields (page 3).

1) Case sensitive file mask pattern allowing wildcards.
2) Secure password.
3) Integer number.
4) Reference to master data.

5) Arbitrary text.

Additional configuration:

System variable	Description
MFT-PKZIP-EXTRACT-PROGRAM	Correct PKUNZIP call. Default value: 'pkzipc –extract -overwrite=all -archivetype=zip -noarchiveextension -embedded=-zip'.
MFT-UNZIP-WORKDIR	Existing working directory. Default value: '$(seeutil:eval('${bisas.data}/../temp/unzip'))'.

Preconditions:

- Script executor adapter started.
- 'mftCallExternalUnzip.bsh' BSH script installed.
- Unzip program from '${MFT-PKZIP-EXTRACT-PROGRAM}' system variable, by default PKZIPC, installed.
- Available process attachment: zip file format.

Result:

- Process attachment: not changed.
  New process attachment: the unpacked file.

Sets the following properties:

- FILE_COUNT := 1. The number of unpacked zip entries and returned attachments.
- FILES_LEFT := 0, in case of success.
- RESULT := 'PKUNZIP'.
- UNCOMPRESS_COUNT := 1. The number of unpacked zip entries.
- UNCOMPRESS_FILESIZE := size of unpacked file.
- UNCOMPRESS_METHOD := 'PKUNZIP'. The method used for uncompression.
- TARGET_NAME := filename of the unpacked file without path.

Possible cases of failure:

Possible case of failure	Success or error behavior
Process attachment without zip format	Error. Throws acUncompress fault.
Empty zip archive	Error. Throws acUncompress fault.
>= 1 file match the zip entry pattern	Error. Throws acUncompress fault.

Policy: none.

1.10.3 acZipList

Caption: ZipList.

Throws Fault=acZipList

The *acZipList* activity is a single file activity that lists the file entries in a zip archive and returns a FILELIST object (cvs file) which can be used as direct input to the *scForEach (page 10)* activity. Uses the *execut eScript* operation of the *ScriptExecutor* component (refer to chapter *2.2.2 executeScript (SeeScriptExecutor)* of the *BIS6 Base Components* manual for more details).

Mode of operation:
Call of a ressource repository BSH script whose name is stored in the ${MFT-RUN-ZIPLIST-SCRIPT} system variable (page 392) with the current process attachment as input attachment (source file). Return the zip entries as output attachment in CSV file format (FILELIST).

Field name	Value	Description
Script parameter		Unused.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data
2) Arbitrary text.

Additional configuration:

System variable	Description
MFT-RUN-ZIPLIST-SCRIPT	Name of the BSH script to be executed.

Preconditions:

- Script Executor adapter started.
- The script from the ${MFT-RUN-ZIPLIST-SCRIPT} system variable available in the ressource repository in '<LS>/scripts/bsh/1' (The ressource repository search uses the current logical system, the BEANSHELL type, and the name '1/<*Script name*>', where '1' is the fix version number).
- Available process attachment: zip file format.

Result:

- New process attachment: resulting zip list as FILELIST object (CSV file).

Sets the following properties:

- LIST_COUNT := The number of entries in the returned list.

Possible cases of failure:

Possible case of failure	Success or error behavior
Empty process attachment	Error. Throws acZipList fault.
Empty zip archive	Error. Throws acZipList fault.
Wrong file format (not zip or corrupted zip)	Error. Throws acZipList fault.
Wrong file format (gzip)	Error. Throws acZipList fault.

Policy: none.

1.11 Security Activities

This section describes security related activities enabling signing and encrypting messages as well as verifying and decrypting messages.

1.11.1 acComposeMIME

Caption: Compose mime.

Throws Fault=acComposeMIME.

The *acComposeMIME* activity is a multiple file activity that composes a MIME or an S/MIME message from the current attachment and a list of bookmarks. Additionally, the resulting message can be digitally signed and/or encrypted. Uses the *processAttachments* operation from the *SMIME Processor* adapter (refer to chapter *1.41.1 processAttachments (link)* of the *BIS6 Base Components* manual or chapter *8.1.1 processAttachments* of the *S/MIME Module* manual for more details).

Field name	Value	Description
Method		Optional. Not editable selection field. Provides available MIME composition modes.
	Compose	Default. Standard composition mode.
Mime type		Mandatory. Non editable selection field. Provides available MIME types.
	application octet	Default. Unspecified data, for instance executable files
	multipart mixed	Multipart message with mixed subtype, no references between the message parts.
	multipart related	Multipart message with related subtype, message parts are dependant from one another, indicating that each one is a component of an aggregate whole. Used for compound objects consisting of several inter-related components - proper display cannot be achieved by individually displaying the constituent parts. One common usage is to send a web page complete with images in a single message.
Boundary	*<Text>*[1]	Optional. Editable text field. Content boundary for multipart messages. This boundary must not occur in any of the message parts. It is placed between the parts as well as at the beginning and at the end of the message body as a kind of message part boundary marker. The *Boundary* supports: • Properties (page 392), e.g. '$[BOUNDARY]'. • System variables (page 392), e.g. '${BOUNDARY}'. • XPath expressions (page 392), e.g. $(upper-case('boundary')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Content type		Mandatory. Non editable selection field. Provides available MIME content types.
	ATTACHMENT	Default. Attachment content disposition. The payload is represented as an attachment in the MIME or S/MIME message.

Field name	Value	Description
	MIME	Mime content disposition. The payload is a complete MIME entity, and not only a body or an attachment, and shall only be signed and/or encrypted. Either *Sign* or *Encrypt* (or both) must be set to 'Alias' or 'From md'. At least one alias has to be set. *Bookmarks* may not be set. Creating headers is not supported for this type.
	PLAIN	Inline content disposition. The payload is represented as a body, or inline attachment, in the MIME or S/MIME message.
Bodypart name	<*Text*>[1]	Optional. Editable text field The standard attachments body part name. The *Bodypart name* supports: • Properties (page 392), e.g. '$[BPNAME]'. • System variables (page 392), e.g. '${BPNAME}'. • XPath expressions (page 392), e.g. $(upper-case('bpName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Bookmarks	<*Bookmark* List>[2]	Optional. Editable text field. Semicolon separated list of bookmark names. These bookmark names are used as names in the MIME message. A differing name can be set using the syntax '<Bookmark>=<DesiredName>'. The *Bookmarks* supports: • Properties (page 392), e.g. '$[BOOKMARK]'. • System variables (page 392), e.g. '${BOOKMARK}'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). • Maximum number of bookmarks is 10. Additional bookmarks are ignored. ⚠ Do not fill this text field if *Content type* is set to 'MIME'.
Bookmark content type		Mandatory. Non editable selection field. Provides available content types for the bookmarks.
	ATTACHMENT	Default. Attachment content disposition.
	MIME	Mime content disposition.
	PLAIN	Inline content disposition.
Encrypt		Mandatory. Non editable selection field. Provides several encryption modes.
	Alias	Default. Use an encryption key that can be directly selected from the keystore. Makes *Encrypt alias* visible.
	From md	Uses the encryption key from the masterdata. Makes *SMIME partner address* visible.

Field name	Value	Description
	No	Does not use an encryption key.
Encrypt alias	*<Ref>*[3]	Optional. Non editable text field with keystore manager selection dialog. The alias of a Keystore Manager certificate. Used for message encryption. Visible when *Encrypt* is set to 'Alias'.
SMIME partner address	*<Ref>*[3]	Optional. Non editable selection field. SMIME masterdata partner address providing the encryption key. Visible when *Encrypt* is set to 'From md'.
Encrypt algorithm		Optional. Provides available symmetric encryption algorithms. Only used if an encrypt alias is given.
	AES_128	Default. Advanced Encryption Standard. Block cipher with 128 bits block size and 128 bits key length.
	AES_192	Advanced Encryption Standard. Block cipher with 128 bits block size and 192 bits key length.
	AES_256	Advanced Encryption Standard. Block cipher with 128 bits block size and 256 bits key length.
	RC2_128	Rivest cipher by Ronald Rivest. Block encryption algorithm with 64 bits block size and 128 bits key length.
	RC2_40	Rivest cipher by Ronald Rivest. Block encryption algorithm with 64 bits block size and 40 bits key length.
	RC2_64	Rivest cipher by Ronald Rivest. Block encryption algorithm with 64 bits block size and 64 bits key length.
	TripleDES_192	Triple Data Encryption Standard. Also referred to as TDES or 3DES. Applies DES cipher algorithm. Block encryption algorithm with192 bits key size.
Sign		Mandatory. Non editable selection field. Provides available signature modes.
	Alias	Default. Use a private signature key that can be directly selected from the keystore. Makes *Signature alias* visible.
	From md	Use a private signature key from SMIME master data. Makes *SMIME own address* visible.
	No	Do not use a private signature key.
Signature alias	*<Ref>*[3]	Optional. Non editable text field with a keystore manager selection dialog. The alias of the private key from the Keystore Manager. Used to sign the message. Visible when *Sign* is set to 'Alias'.
SMIME own address	*<Ref>*[3]	Mandatory. Editable selection dialog. SMIME masterdata own address with a private signature key. The *SMIME own address* supports: • Properties (page 392), e.g. '$[SMIME_OWN_ADDRESS_ID]'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')) System variables are not supported. Visible when *Sign* is set to 'From md'.

Field name	Value	Description
Signing algorithm		Optional. Non editable selection field. Provides available hash algorithms used to sign the message. Only used if a signature alias is given.
	MD5	Default. Message-digest algorithm 5. Produces a 128-bit hash value. Aimed at 32-bit machines.
	SHA1	Secure hash algorithm. SHA-1 hash function. Produces a 160-bit hash value.
Add certificate chain	True	The used certificate chain is included into the resulting message. It is added to the signature. It is ignored if *Sign* is set to 'No'.
	False	Default. Nothing is added to the signature.
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary Text.
2) Semicolon separated list of bookmark names i.e. arbitrary names except invalid characters ", &, < (which cannot be inserted).
3) Reference to master data.

Preconditions:

- S/MIME module adapter started.
- Available process attachment.
- Valid bookmarked attachments.

Result:

- New process attachment: composed MIME or S/MIME message.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 'Composed'.
- DECRYPTION_METHOD := 'MIME'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid bookmark in *Bookmarks*	Error. Throws acComposeMIME fault.
Invalid property in editable text field	Error. Throws acComposeMIME fault.
Invalid system variable in editable text field	Error. Throws acComposeMIME fault.
MIME content type without signature or encryption key	Error. Throws acComposeMIME fault.
MIME content type with one or more bookmarks	Error. Throws acComposeMIME fault.
Key of signature alias is not private	Error. Throws acComposeMIME fault.

Policy:

ALLOWANCE/PAYLOAD-DECRYPT

1.11.2 acDecomposeMIME

Caption: Decompose mime.

Throws Fault=acDecomposeMIME.

The *acDecompose* activity decomposes the current mime message. It returns either the first attachment or a named attachment from the mime message. Uses the *processAttachments* operation from the *SMIME Processor* adapter (refer to chapter *1.41.1 processAttachments (link)* of the *BIS6 Base Components* manual or chapter *8.1.1 processAttachments* of the *S/MIME Module* manual for more details).

Field name	Value	Description
Method		Mandatory. Non editable selection field. Provides available MIME decomposition methods.
	Get first attachment of mime	Default. Get the first attachment of the mime message.
	Get specific attachment from mime	Get a named attachment of the mime message. Makes the *Bodypart name* visible.
Bodypart name	*<Text>*[1]	Optional. Editable text field. The standard attachments body part name. The Bodypart name supports: • The *Bodypart name* supports: • Properties (page 392), e.g. '$[BPNAME]'. • System variables (page 392), e.g. '${BPNAME}'. • XPath expressions (page 392), e.g. $(upper-case('bpName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible for the 'Get specific attachment' *Method* only.
Validate		Mandatory. Non editable selection field. Provides available validation modes.
	Alias	Default. Uses a validation key that can be directly selected from the keystore. Makes *Validation alias* visible.
	From md	Uses the validation key from the masterdata. Makes *SMIME partner address* visible.
	No	Does not use a validation key.
Validation alias	*<Ref>*[2]	Optional. Non editable text field with a keystore manager selection dialog. The alias of the private key from the Keystore Manager.
SMIME partner address	*<Ref>*[2]	Optional. Non editable selection field. SMIME masterdata partner address with a private signature key. Visible when *Validate* is set to 'From md'.
Decrypt		Mandatory. Non editable selection field. Provides available decryption modes.
	Alias	Default. Uses a decryption key that can be directly selected from the keystore. Makes *Decryption alias* visible.
	From md	Uses the decryption key from the masterdata.

Field name	Value	Description
		Makes *SMIME own address* visible.
	No	Does not use any decryption key.
Decryption alias	*<Ref>*[2]	Optional. Non editable text field with a keystore manager selection dialog. The alias of the private key from the Keystore Manager.
SMIME own address	*<Ref>*[2]	Optional. Non editable selection field. SMIME masterdata own address with a private decryption key. Visible when *Decrypt* is set to 'From md'.
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary Text.
2) Reference to master data.

Preconditions:

- S/MIME module adapter started.

Results:

- MIME message parts as attachments

The following properties are set:

- STATUS := 'SUCCESS.
- RESULT := 100.
- MimeBodypartName := 'attachment-<payload@refID>.dat'
- MimeBodypartPosition := 'FIRST' (for 'Get first attachment of mime' method) or 'unknown' (for 'Get specific attachment from mime' method).

Possible cases of failure:

Possible case of failure	Success or error behavior
Arbitrary text file, no MIME format	Success. Empty result file.

Policy:

ALLOWANCE/PAYLOAD-DECRYPT

1.11.3 acMIMEList

Caption: Compose mime.

Throws Fault=acMimeList.

The *acMIMEList* activity lists the content of the current MIME message. Returns all body parts as bookmarks. Uses the *processAttachments* operation from the *SMIME Processor* adapter (refer to chapter *1.41.1 processAttachments (link)* of the *BIS6 Base Components* manual or chapter *8.1.1 processAttachments (link)* of the *S/MIME Module* manual for more details).

Mode of operation:

The *acMIMEList* activity extracts all message parts from the MIME message in the current process attachment. This list of mime message entries is converted into an output attachment in CSV format (FILELIST) by the *BIC Client* adapter and can be used as input for the *scForEach* (page 10) looping activity.

Field name	Value	Description
Mapping parameter	*<Text>*[1]	Optional. Editable text field. Parameter (value) for the mapping used to create the resulting file list. Handed to the BIC Client adapter using the name 'PARAMETER'. The name of the standard mapping is fetched from the system variable (page 392) ${MFT-FILELISTCREATOR} which by default does not use any mapping parameters. The *Mapping parameter* supports: • Properties (page 392), e.g. '$[PARAM_VALUE]'. • System variables (page 392), e.g. '${PARAM_VALUE}'. • XPath expressions (page 392), e.g. $(upper-case('paramvalue')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Allow in depth multipart parsing	True	Decompose the complete S/MIME message into atomic parts. Return a list containing all parts.
	False	Parse only the top layer of the MIME message. Create a list containing only the top-layer parts.
Validation alias	*<Ref>*[2]	Optional. Non editable text field with keystore manager selection dialog. Alias of a key from the Keystore Manager.
Decryption alias	*<Ref>*[2]	Optional. Non editable text field with keystore manager selection dialog. Alias of a private decryption key from the Keystore Manager.
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.
2) Reference to master data.

Additional configuration:

System variable	Description
MFT-FILELISTCREATOR	Name of mapping for conversion of list of mime message entries into FILELIST CSV format. ⚠ Changing this mapping affects all list activities with FILELIST format!

Preconditions:

- S/MIME module adapter started.
- BIC Client adapter started.
- Mapping from ${MFT-FILELISTCREATOR} system variable available in repository.
- Available process attachment: mime message format.

Results:

- New process attachment: resulting Mime entries list as FILELIST object (CSV file).

The following properties are set:

- <BodypartName> := XML response schema of the body part of the MIME message.
- <BookmarkName> := XML response scheme of additional message part, either name of used bookmark, or file name assigned to the bookmark name.
- LIST_COUNT := Number of message parts.

Possible cases of failure:

Possible case of failure	Success or error behavior
No validation alias and no encryption alias for signed and excrypted MIME message	Success. No validation, no encryption. Returns 1 entry with the type 'BOOKMARK' and name 'smime.p7m'.
Invalid system variable in *Mapping parameter*	Error. Throws acMIMEList fault.
Invalid property in *Mapping parameter*	Success. Uses unresolved value for PARAMETER.

Policy:

ALLOWANCE/PAYLOAD-DECRYPT

1.11.4 acEncrypt

Caption: PGPEncrypt.

Throws Fault=acEncrypt.

The *acEncrypt* activity is a single file activity that signs and/or encrypts the current process attachment. Additional features include compressing and changing the destination queue for 'big' files. Uses the *Sign*, *Encrypt*, and *SignEncrypt*operations of the *OpenPGP Module* component.

Field name	Value	Description
Method		Mandatory. Non-editable selection field. Provides supported signing and encryption methods as well as combinations.
	Sign with key	Default. Sign the current process attachment with a PGP secret key from the keystore manager.
	Encrypt with key	Encrypt the current process attachment with a PGP public key from the keystore manager.
	Encrypt with password	Encrypt the current process attachment with a password.
	Sign and encrypt with key	Sign and encrypt the current process attachment with PGP keys from the keystore manager.
	Sign with key encrypt with passphrase	Sign the current process attachment with a PGP secret key and encrypt it with a passphrase.

Additional configuration:

System variable	Description
MFT-PGP-THRESHOLD-SIZE	Default value 100 MB (100.000.000B). Threshold for file size. Needed for special handling of 'big' files.
MFT-PGP-BIG-FILE-DESTINATION	Default value 'sync:'. Destination queue for signing and/or encrypting if the current process attachment is bigger than the MFT-PGP-THRESHOLD-SIZE system variable. Destination queue for handling smaller PGP documents is always 'sync:'.

System variable	Description
	Possible use case is specification of a destination queue on another node for 'big' files.

Method: Sign with key

This method signs the current process attachment with a PGP secret key from the keystore manager. Uses the *Sign* operation of the *OpenPGP Module* component (refer to chapter *10.1.2 Sign* of the *OpenPGP Module* manual for more details).

Field name	Value	Description
Filename	*<Filename>*[1]	Optional. Editable text field. File name to use for the signed document. If empty, the attachment ID is used. The *Filename* supports: • System variables (page 392), e.g. '${FILE_NAME}'. • Properties (page 392), e.g. '$[FILE_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('fileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Pay attention when entering file names. Because of XPath expression support it is possible to enter invalid file names such as file names consisting, for instance, of asterisks or blanks. If the file name consists of 1 or more blanks only, validating the signed file with the *acDecrypt* (page 171) activitiy results in a TARGET_NAME property with the value 'NOT_SET'.
ASCII armor	True	Radix-64 encoding of message. Refer to chapter *12.1 ASCII Armoring* in the *OpenPGP Module* manual for more details.
	False	Default. No additional encoding.
Compression		Optional. Non-editable selection field. Provides several compression methods.
	UNCOMPRESSED	Default. Do not use any compression. Refer to chapter *12.2 Compression* in the *OpenPGP Module* manual for more details.
	ZIP	Use ZIP compression.
	ZLIB	Use zlib data compression.
	BZIP2	Use bzip2 compression.
Sign alias	*<Ref>*[2]	Optional. Non-editable text field with keystore manager selection dialog. PGP secret key from the Keystore Manager 'USERS/PGP', required for signing action.
Key passphrase (sign)	*<Passphrase>*[3]	Optional. Editable password field. Pass phrase protecting the PGP secret key for signing. If this is empty the pass phrase is taken from the KSM key configuration if possible. Properties, system variables and XPaths are not supported.
Hash algorithm		Mandatory. Non-editable selection field. Provides available hash algorithms.
	SHA-1	Default. Secure hash algorithm. SHA-1 hash function. Produces a 160-bit hash value.
	SHA-224	Secure hash algorithm. SHA-2 hash function. Produces a 224-bit hash value.

Field name	Value	Description
	SHA-256	Secure hash algorithm. SHA-2 hash function. Produces a 256-bit hash value.
	SHA-384	Secure hash algorithm. SHA-2 hash function. Produces a 384-bit hash value.
	SHA-512	Secure hash algorithm. SHA-2 hash function. Produces a 512-bit hash value.
	MD2	Message-digest algorithm 2. Produces a 128-bit hash value. Optimized for 8-bit machines.
	MD5	Message-digest algorithm 5. Produces a 128-bit hash value. Aimed at 32-bit machines.
	RIPE-MD-160	Race Integrity Primitives Evaluation Message Digest. Produces a 160-bit hash value.
One pass signature	True	Compute entire signed message in one pass.
	False	Default. Creates a PGP 2.x compliant message. Refer to chapter *11 Compatibility Considerations* in the *OpenPGP Module* manual for more details.
Document type		Non-editable selection field. Provides 2 document types.
	BINARY	Default. Any file.
	CANONICAL_ TEXT	Text file containing printable characters and lines end with CRLF.
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Secure passphrase.
4) Arbitrary text.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- PGP secret key for sign alias.
- Current process attachment.

Result:

- New process attachment: signed and possibly compressed message.

Sets the following properties:

- ENCRYPT_METHOD := 'SIGNED'.
- TARGET_NAME := The file name for the signed document. If not specified, the used attachment ID.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acEncrypt fault.
No sign alias	Error. Throws acEncrypt fault.
PGP key for alias not found	Error. Throws acEncrypt fault.
PGP key for alias renamed or store renamed: key not found or wrong key found	Error. Throws acEncrypt fault.
Wrong password for sign alias	Error. Throws acEncrypt fault.
Public key for sign alias	Error. Throws acEncrypt fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP

Method: Encrypt with key

This method encrypts the current process attachment with a PGP public key from the keystore manager using a symmetric encryption algorithm. Uses the *Encrypt* operation of the *OpenPGP Module* component (refer to chapter *10.1.3 Encrypt* of the *OpenPGP Module* manual for more details).

Field name	Value	Description
Filename	*<Filename>*[1]	Optional. Editable text field. File name to use for the signed document. If empty, the attachment ID is used. The *Filename* supports: • System variables (page 392), e.g. '${FILE_NAME}'. • Properties (page 392), e.g. '$[FILE_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('fileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Pay attention when entering file names. Because of XPath expression support it is possible to enter invalid file names such as file names consisting, for instance, of asterisks or blanks. If the file name consists of 1 or more blanks only, decrypting the encrypted file with the *acDecrypt* (page 171) activitiy results in a TARGET_NAME property with the value 'NOT_SET'.
ASCII armor	True	Radix-64 encoding of message. Refer to chapter *12.1 ASCII Armoring* in the *OpenPGP Module* manual for more details.
	False	Default. No additional encoding.
Compression		Mandatory. Non-editable selection field. Provides several compression methods
	UNCOMPRESSED	Default. Do not use any compression. Refer to chapter *12.2 Compression* in the *OpenPGP Module* manual for more details.
	ZIP	Use ZIP compression.
	ZLIB	Use zlib data compression.
	BZIP2	Use bzip2 compression.
Document type		Mandatory. Non-editable selection field. Provides 2 document types.
	BINARY	Default. Any file.

Field name	Value	Description
	CANONICAL_ TEXT	Text file containing printable characters and lines end with CRLF.
Encrypt alias	*<Ref>*[2]	Optional. Non-editable text field with keystore manager selection dialog. PGP public key from the Keystore Manager 'TRUSTED/PGP', required for encryption. Example: 'Partner certificates(TRUSTED\PGP\encryptdecrypt'.
Symmetric algorithm		Mandatory. Non-editable selection field. Provides available symmetric cryptographic algorithms. Refer to chapter *12.3 Encryption Algorithms* in the *OpenPGP Module* manual for more details.
	AES-128	Default. Advanced Encryption Standard. Block cipher with 128 bits block size and 128 bits key length.
	AES-192	Advanced Encryption Standard. Block cipher with 128 bits block size and 192 bits key length.
	BLOWFISH	Block cipher with 64 bits block size and variable key length from 32 bits up to 448 bits.
	CAST5	Also referred to as CAST-128. Block cipher with 64 bits block size and variable key length from 40 bits up to 128 bits.
	IDEA	International Data Encryption Algorithm. Block cipher with 64 bits block size and 128 bits key length.
	SAFER-SK128	Secure And Fast Encryption Routine with Strengthened Key schedule. Block cipher with 64 bits block size and 128 bits key length.
	TRIPLEDES	Also referred to as TDES or 3DES. Triple Data Encryption Standard. Applies DES cipher algorithm.
Ignore integrity check	True	Omit integrity check.
	False	Default. Add integrity check MDC (mofdification detection code). Refer to chapter *12.2 Compression* in the *OpenPGP Module* manual for more details.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- PGP public key for encrypt alias.
- Current process attachment: binary data or text file.

Result:

- New process attachment: encrypted PGP document, possibly compressed.

Sets the following properties:

- ENCRYPT_METHOD := 'ENCRYPTED'.
- TARGET_NAME := The file name for the encrypted document.

'NOT_SET' if the file name entered consists of 1 or more blanks only.
The attachment ID, if the file name is not specified.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acEncrypt fault.
No encrypt alias	Error. Throws acEncrypt fault.
PGP key for alias renamed or store renamed: key not found or wrong key found	Error. Throws acEncrypt fault.
Secret key for encrypt alias	Error. Throws acEncrypt fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP

Method: Encrypt with password

This method encrypts the current process attachment with a password. Uses the *Encrypt* operation of the *OpenPGP Module* component (refer to chapter *10.1.3* t of the *OpenPGP Module* manual for more details).

Field name	Value	Description
Filename	*<Filename>*[1]	Optional. Editable text field. File name to use for the signed document. If empty, the attachment ID is used. The *Filename* supports: • System variables (page 392), e.g. '${FILE_NAME}'. • Properties (page 392), e.g. '$[FILE_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('fileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Pay attention when entering file names. Because of XPath expression support it is possible to enter invalid file names such as file names consisting, for instance, of asterisks or blanks. If the file name consists of 1 or more blanks only, decrypting the encrypted file with the *acDecrypt* (page 171) activitiy results in a TARGET_NAME property with the value 'NOT_SET'.
ASCII armor	True	Radix-64 encoding of message. Refer to chapter *12.1 ASCII armoring* in the *OpenPGP Module* manual for more details.
	False	Default. No additional encoding.
Compression		Mandatory. Non-editable selection field. Provides several compression methods.
	UNCOMPRESSED	Default. Do not use any compression. Refer to chapter *12.2 Compression* in the *OpenPGP Module* manual for more details.
	ZIP	Use ZIP compression.
	ZLIB	Use zlib data compression.
	BZIP2	Use bzip2 compression.
Password (encrypt)	*<Password>*[2]	Mandatory. Editable password field. Encryption password.

Field name	Value	Description
		Properties, system variables and XPaths are not supported.
Password encryption algorithm		Mandatory. Non-editable selection field. Provides available symmetric encryption algorithms. Refer to chapter *12.3 Encryption Algorithms* in the *OpenPGP Module* manual for more details.
	AES-128	Default. Advanced Encryption Standard. Block cipher with 128 bits block size and 128 bits key length.
	AES-192	Advanced Encryption Standard. Block cipher with 128 bits block size and 192 bits key length.
	BLOWFISH	Block cipher with 64 bits block size and variable key length from 32 bits up to 448 bits.
	CAST5	Also referred to as CAST-128. Block cipher with 64 bits block size and variable key length from 40 bits up to 128 bits.
	IDEA	International Data Encryption Algorithm. Block cipher with 64 bits block size and 128 bits key length.
	SAFER-SK128	Secure And Fast Encryption Routine with Strengthened Key schedule. Block cipher with 64 bits block size and 128 bits key length.
	TRIPLEDES	Also referred to as TDES or 3DES. Triple Data Encryption Standard. Applies DES cipher algorithm.
Document type		Non-editable selection field. Provides 2 document types.
	BINARY	Default. Any file.
	CANONICAL_TEXT	Text file containing printable characters and lines end with CRLF.
Symetric algorithm		Mandatory. Non-editable selection field. Provides available symmetric encryption algorithms. Refer to chapter *12.3 Encryption Algorithms* in the *OpenPGP Module* manual for more details.
	AES-128	Default. Advanced Encryption Standard. Block cipher with 128 bits block size and 128 bits key length.
	AES-192	Advanced Encryption Standard. Block cipher with 128 bits block size and 192 bits key length.
	BLOWFISH	Block cipher with 64 bits block size and variable key length from 32 bits up to 448 bits.
	CAST5	Also referred to as CAST-128. Block cipher with 64 bits block size and variable key length from 40 bits up to 128 bits.
	IDEA	International Data Encryption Algorithm. Block cipher with 64 bits block size and 128 bits key length.
	SAFER-SK128	Secure And Fast Encryption Routine with Strengthened Key schedule. Block cipher with 64 bits block size and 128 bits key length.
	TRIPLEDES	Also referred to as TDES or 3DES. Triple Data Encryption Standard. Applies DES cipher algorithm.
Ignore integrity check	True	Omit integrity check.
	False	Default. Add integrity check MDC (mofdification detection code). Refer to chapter *11 Compatibility Considerations* in the *OpenPGP Module* manual for more details.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Secure password.
3) Arbitrary text.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- Current process attachment.

Result:

- New process attachment: encrypted and possibly compressed message.

Sets the following properties:

- ENCRYPT_METHOD := 'ENCRYPTED'.
- TARGET_NAME := The file name for the encrypted document.
  'NOT_SET' if the file name entered consists of 1 or more blanks only.
  The attachment ID, if the file name is not specified.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acEncrypt fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP

Method: Sign and encrypt with key

This method signs and encrypts the current process attachment with PGP keys from the keystore manager. Uses the *SignEncrypt* operation of the *OpenPGP Module* component (refer to chapter *10.1.4 Sign Encrypt* of the *OpenPGP Module* manual for more details).

Field name	Value	Description
Filename	*<Filename>*[1]	Optional. Editable text field. File name to use for the signed document. If empty, the attachment ID is used. The *Filename* supports: • System variables (page 392), e.g. '${FILE_NAME}'. • Properties (page 392), e.g. '$[FILE_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('fileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Pay attention when entering file names. Because of XPath expression support it is possible to enter invalid file names such as file names consisting, for instance, of asterisks or blanks. If the file name consists of 1 or more blanks only, validating/ decrypting the signed and encrypted file with the *acDecrypt* (page 171) activitiy results in a TARGET_NAME property with the value 'NOT_SET'.

Field name	Value	Description
ASCII armor	True	Radix-64 encoding of message. Refer to chapter *12.1 ASCII armoring* in the *OpenPGP Module* manual for more details.
	False	Default. No additional encoding.
Compression		Mandatory. Non-editable selection field. Provides several compression methods.
	UNCOMPRESSED	Default. Do not use any compression. Refer to chapter *12.2 compression* in the *OpenPGP Module* manual for more details.
	ZIP	Use ZIP compression.
	ZLIB	Use zlib data compression.
	BZIP2	Use bzip2 compression.
Sign alias	*<Ref>*[2]	Optional. Non-editable text field with keystore manager selection dialog. PGP secret key from keystore manager 'USERS/PGP', required for signing action.
Key passphrase (sign)	*<Passphrase>*[3]	Optional. Editable password field. Pass phrase protecting the PGP secret key for signing. If this is empty the pass phrase is taken from the KSM key configuration if possible. Properties, system variables and XPaths are not supported.
Hash algorithm		Mandatory. Non-editable selection field. Provides available hash algorithms.
	SHA-1	Default. Secure hash algorithm. SHA-1 hash function. Produces a 160-bit hash value.
	SHA-224	Secure hash algorithm. SHA-2 hash function. Produces a 224-bit hash value.
	SHA-256	Secure hash algorithm. SHA-2 hash function. Produces a 256-bit hash value.
	SHA-384	Secure hash algorithm. SHA-2 hash function. Produces a 384-bit hash value.
	SHA-512	Secure hash algorithm. SHA-2 hash function. Produces a 512-bit hash value.
	MD2	Message-digest algorithm 2. Produces a 128-bit hash value. Optimized for 8-bit machines.
	MD5	Message-digest algorithm 5. Produces a 128-bit hash value. Aimed at 32-bit machines.
	RIPE-MD-160	Race Integrity Primitives Evaluation Message Digest. Produces a 160-bit hash value.
One pass signature	True	Compute entire signed message in one pass.
	False	Default. Creates a PGP 2.x compliant message. Refer to chapter *11 Compatibility Considerations* in the *OpenPGP Module* manual for more details.
Document type		Non-editable selection field. Provides 2 document types.
	BINARY	Default. Any file.
	CANONICAL_TEXT	Text file containing printable characters and lines end with CRLF.
Encrypt alias	*<Ref>*[2]	Optional. Non-editable text field with keystore manager selection dialog. PGP public key from the Keystore Manager 'TRUSTED/PGP', required for encryption.

Field name	Value	Description
Symetric algorithm		Optional. Non-editable selection field. Provides available symmetric encryption algorithms. Refer to chapter *12.3 Encryption Algorithms* in the *OpenPGP Module* manual for more details.
	AES-128	Default. Advanced Encryption Standard. Block cipher with 128 bits block size and 128 bits key length.
	AES-192	Advanced Encryption Standard. Block cipher with 128 bits block size and 192 bits key length.
	BLOWFISH	Block cipher with 64 bits block size and variable key length from 32 bits up to 448 bits.
	CAST5	Also referred to as CAST-128. Block cipher with 64 bits block size and variable key length from 40 bits up to 128 bits.
	IDEA	International Data Encryption Algorithm. Block cipher with 64 bits block size and 128 bits key length.
	SAFER-SK128	Secure And Fast Encryption Routine with Strengthened Key schedule. Block cipher with 64 bits block size and 128 bits key length.
	TRIPLEDES	Also referred to as TDES or 3DES. Triple Data Encryption Standard. Applies DES cipher algorithm.
Ignore integrity check	True	Omit integrity check.
	False	Default. Add integrity check MDC (mofdification detection code). Refer to chapter *11 Compatibility Considerations* in the *OpenPGP Module* manual for more details.
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Secure passphrase.
4) Arbitrary text.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- PGP secret key for sign alias.
- PGP public key for encrypt alias.
- Current process attachment.

Result:

- New process attachment: encrypted and possibly compressed message.

Sets the following properties:

- ENCRYPT_METHOD := 'SIGNED;ENCRYPTED'.
- TARGET_NAME := The file name for the signed and encrypted document.
  'NOT_SET' if the file name entered consists of 1 or more blanks only.
  The attachment ID, if the file name is not specified.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acEncrypt fault.
No sign alias	Error. Throws acEncrypt fault.
No encrypt alias	Error. Throws acEncrypt fault.
PGP key for alias not found	Error. Throws acEncrypt fault.
PGP key for alias renamed or store renamed: key not found or wrong key found	Error. Throws acEncrypt fault.
Public key for sign alias	Error. Throws acEncrypt fault.
Secret key for encrypt alias	Error. Throws acEncrypt fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP

Method: Sign with key encrypt with passphrase

This method signs the current process attachment with a PGP secret key from the keystore manager and encrypts it with a passphrase. Uses the *SignEncrypt* operation of the *OpenPGP Module* component (refer to chapter *10.1.4 SignEncrypt* of the *OpenPGP Module* manual for more details).

Field name	Value	Description
Filename	*<Filename>*[1]	Optional. Editable text field. File name to use for the signed document. If empty, the attachment ID is used. The *Filename* supports: • System variables (page 392), e.g. '${FILE_NAME}'. • Properties (page 392), e.g. '$[FILE_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('fileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Pay attention when entering file names. Because of XPath expression support it is possible to enter invalid file names such as file names consisting, for instance, of asterisks or blanks. If the file name consists of 1 or more blanks only, validating/decrypting the signed and encrypted file with the *acDecrypt* (page 171) activitiy results in a TARGET_NAME property with the value 'NOT_SET'.
ASCII armor	True	Radix-64 encoding of message. Refer to chapter *12.1 ASCII armoring* in the *OpenPGP Module* manual for more details.
	False	Default. No additional encoding.
Compression		Mandatory. Non-editable selection field. Provides several compression methods.
	UNCOMPRESSED	Default. Do not use any compression. Refer to chapter *12.2 Compression* in the *OpenPGP Module* manual for more details.
	ZIP	Use ZIP compression.
	ZLIB	Use zlib data compression.
	BZIP2	Use bzip2 compression.

Field name	Value	Description
Sign alias	*<Ref>*[2]	Optional. Non-editable text field with keystore manager selection dialog. PGP secret key from keystore manager 'USERS/PGP', required for signing action.
Key passphrase (sign)	*<Passphrase>*[3]	Optional. Editable password field. Pass phrase protecting the PGP secret key for signing. Properties, system variables and XPaths are not supported.
Password (encrypt)	*<Password>*[4]	Mandatory. Editable password field. Encryption password. Properties, system variables and XPaths are not supported.
Password encryption algorithm		Mandatory. Non-editable selection field.
	AES-128	Default. Advanced Encryption Standard. Block cipher with 128 bits block size and 128 bits key length.
	AES-192	Advanced Encryption Standard. Block cipher with 128 bits block size and 192 bits key length.
	BLOWFISH	Block cipher with 64 bits block size and variable key length from 32 bits up to 448 bits.
	CAST5	Also referred to as CAST-128. Block cipher with 64 bits block size and variable key length from 40 bits up to 128 bits.
	IDEA	International Data Encryption Algorithm. Block cipher with 64 bits block size and 128 bits key length.
	SAFER-SK128	Secure And Fast Encryption Routine with Strengthened Key schedule. Block cipher with 64 bits block size and 128 bits key length.
	TRIPLEDES	Also referred to as TDES or 3DES. Triple Data Encryption Standard. Applies DES cipher algorithm.
Hash algorithm		Mandatory. Non-editable selection field. Provides available hash algorithms.
	SHA-1	Default. Secure hash algorithm. SHA-1 hash function. Produces a 160-bit hash value.
	SHA-224	Secure hash algorithm. SHA-2 hash function. Produces a 224-bit hash value.
	SHA-256	Secure hash algorithm. SHA-2 hash function. Produces a 256-bit hash value.
	SHA-384	Secure hash algorithm. SHA-2 hash function. Produces a 384-bit hash value.
	SHA-512	Secure hash algorithm. SHA-2 hash function. Produces a 512-bit hash value.
	MD2	Message-digest algorithm 2. Produces a 128-bit hash value. Optimized for 8-bit machines.
	MD5	Message-digest algorithm 5. Produces a 128-bit hash value. Aimed at 32-bit machines.
	RIPE-MD-160	Race Integrity Primitives Evaluation Message Digest. Produces a 160-bit hash value.
One pass signature	True	Compute entire signed message in one pass.
	False	Default. Creates a PGP 2.x compliant message. Refer to chapter *11 Compatibility Considerations* in the *OpenPGP Module* manual for more details.

Field name	Value	Description
Document type		Non-editable selection field. Provides 2 document types.
	BINARY	Default. Any file.
	CANONICAL_ TEXT	Text file containing printable characters and lines end with CRLF.
Symmetric algorithm		Optional. Non-editable selection field. Provides available symmetric encryption algorithms. Refer to chapter *12.3 Encryption Algorithms* in the *OpenPGP Module* manual for more details.
	AES-128	Default. Advanced Encryption Standard. Block cipher with 128 bits block size and 128 bits key length.
	AES-192	Advanced Encryption Standard. Block cipher with 128 bits block size and 192 bits key length.
	BLOWFISH	Block cipher with 64 bits block size and variable key length from 32 bits up to 448 bits.
	CAST5	Also referred to as CAST-128. Block cipher with 64 bits block size and variable key length from 40 bits up to 128 bits.
	IDEA	International Data Encryption Algorithm. Block cipher with 64 bits block size and 128 bits key length.
	SAFER-SK128	Secure And Fast Encryption Routine with Strengthened Key schedule. Block cipher with 64 bits block size and 128 bits key length.
	TRIPLEDES	Also referred to as TDES or 3DES. Triple Data Encryption Standard. Applies DES cipher algorithm.
Log message	*<Text>*[5]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Secure passphrase.
4) Secure password.
5) Arbitrary text.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- PGP secret key for sign alias.
- Current process attachment.

Result:

- New process attachment: encrypted and possibly compressed message.

Sets the following properties:

- ENCRYPT_METHOD := 'SIGNEDWITHKEY;ENCRYPTEDWITHPASSPHRASE'.
- TARGET_NAME := The file name for the signed and encrypted document.
  'NOT_SET' if the file name entered consists of 1 or more blanks only.
  The attachment ID, if the file name is not specified.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acEncrypt fault.
No sign alias	Error. Throws acEncrypt fault.
PGP key for alias not found	Error. Throws acEncrypt fault.
PGP key for alias renamed or store renamed: key not found or wrong key found	Error. Throws acEncrypt fault.
Wrong password for sign alias	Error. Throws acEncrypt fault.
Public key for sign alias	Error. Throws acEncrypt fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP

1.11.5 acDecrypt

Caption: PGPDecrypt.

Throws Fault=acDecrypt.

The *acDecrypt* activity is a single file activity that verifies the signature and/or decrypts the current process attachment, expecting a PGP document. Additional features include changing the destination queue for 'big' files. Uses the Verify, Decrypt, and DecryptVerify operations of the *OpenPGP Module* component.

Field name	Value	Description
Method		Mandatory. Non-editable selection field. Provides several methods for signature verification and PGP document deccryption.
	Verify	Default. Verifies the signature of the current process attachment (PGP document) with a PGP key.
	Decrypt with key	Decrypts the current process attachment (PGP document) with a passphrase encrypted PGP key.
	Decrypt with password	Decrypts the current process attachment (PGP document) with a password.
	Decrypt and verify with key	Decrypts and verifies the current process attachment (PGP document) with PGP keys.
	Verify with key and decrypt with passphrase	Decrypts the current process attachment (PGP document) with a password and verifies its signature with a PGP key.
	Decrypt with key and no verify (auto)	Decrypts the current process attachment (PGP document) with a PGP key and does not verify its signature.

Additional configuration:

System variable	Description
MFT-PGP-THRESHOLD-SIZE	Default value 100 MB (100.000.000B). Threshold for file size. Needed for special handling of 'big' files.
MFT-PGP-BIG-FILE-DESTINATION	Default value 'sync:'. Destination queue for verifying and/or decrypting if the current process attachment is bigger than the MFT-PGP-THRESHOLD-SIZE system variable. Destination queue for handling smaller PGP documents is always 'sync:'.

System variable	Description
	Possible use case is specification of a destination queue on another node for 'big' files.

Method: Verify

This method verifies the current process attachment with a PGP public key from the keystore manager. Uses the *Verify* operation of the *OpenPGP Module* component (refer to chapter *10.3.2 Verify* of the *OpenPGP M odule* manual) for more details).

Field name	Value	Description
Verify alias	*<Ref>*[2]	Optional. Non-editable text field with Keystore Manager selection dialog. The PGP public key required by the verification.
Abort on error	True/False	Refer to Common Fields.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- Current process attachment: PGP document.

Result:

- New process attachment: If a detached signature has been verified, the message loaded from *detachedSignature attachmentRef.*

Sets the following properties:

- DECRYPT_METHOD := 'VERIFIED'.
- TARGET_NAME := File name of the message extracted from the signed PGP document. 'NOT_SET' if no file name exists.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acDecrypt fault.
Message is no PGP document	Error. Throws acDecrypt fault.
Message is an encrypted PGP document	Error. Throws acDecrypt fault.
No alias	Error. Throws acDecrypt fault.
Signature key is not matching	Error. Throws acDecrypt fault.
Referenced PGP key in key store renamed: key not found or wrong key found	Error. Throws acDecrypt fault.
PGP document without signature	Error. Throws acDecrypt fault.

Policy:

ALLOWANCE/PAYLOAD-DECRYPT

Method: Decrypt with key

This method decrypts the current process attachment with a PGP private key from the keystore manager. Uses the *Decrypt* operation of the *OpenPGP Module* component (refer to chapter *10.3.3 Decrypt (link)* of the *OpenPGP Module* manual) for more details).

Field name	Value	Description
Decrypt alias	*<Ref>*[1]	Optional. Non-editable text field with Keystore Manager selection dialog. The PGP private key required by the decryption.
Key passphrase (decrypt)	*<Passphrase>*[2]	Optional. Editable password field. The passphrase required by decryption. Properties, system variables and XPath expressions not supported.
Abort on error	True/False	Refer to Common Fields.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Secure passphrase.
3) Arbitrary text.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- Current process attachment: PGP document.

Result:

- New process attachment: decrypted message.

Sets the following properties:

- DECRYPT_METHOD := 'DECRYPTED'.
- TARGET_NAME := File name of the message extracted from the PGP document. 'NOT_SET' if no file name exists.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acDecrypt fault.
Message is no PGP document	Error. Throws acDecrypt fault.
Decryption key is not matching	Error. Throws acDecrypt fault.
Passphrase is wrong	Error. Throws acDecrypt fault.

Policy:

ALLOWANCE/PAYLOAD-DECRYPT

Method: Decrypt with password

This method decrypts the current process attachment. Uses the *Decrypt* operation of the *OpenPGP Module* component (refer to chapter *10.3.3 Decrypt (link)* of the *OpenPGP Module* manual) for more details).

Field name	**Value**	**Description**
Password	*<Password>*[1]	???????? Optional. Editable password field. The password required by decryption.
Abort on error	True/False	Refer to Common Fields.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Secure password.
2) Arbitrary text.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- Current process attachment: PGP document.

Result:

- New process attachment: decrypted message.

Sets the following properties:

- DECRYPT_METHOD := 'DECRYPTED'.
- TARGET_NAME := File name of the message extracted from the PGP document or attachment ID if no file name exists.

Possible cases of failure:

Possible case of failure	**Success or error behavior**
OpenPGP Module adapter not started	Error. Throws acDecrypt fault.
Message is no PGP document	Error. Throws acDecrypt fault.
Password is wrong	Error. Throws acDecrypt fault.

Policy:

ALLOWANCE/PAYLOAD-DECRYPT

Method: Decrypt and verify with key

This method verifies and decrypts the current process attachment. Uses the *DecryptVerify* operation of the *OpenPGP Module* component (refer to chapter *10.3.4 DecryptVerify* of the manual) for more details).

Field name	**Value**	**Description**
Decrypt alias	*<Ref>*[1]	Optional. Non-editable text field with Keystore Manager selection dialog. The PGP private key required by decryption.
Key passphrase (decrypt)	*<Passphrase>*[2]	Optional. Editable password field. The passphrase required by decryption. Properties, system variables and XPath expressions not supported.

Field name	Value	Description
Verify alias	*<Ref>*[1]	Optional. Non-editable text field with Keystore Manager selection dialog. The PGP public key required by verification.
Abort on error	True/False	Refer to Common Fields.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Secure passphrase.
3) Arbitrary text.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- Current process attachment: PGP document.

Result:

- New process attachment: decrypted message.

Sets the following properties:

- DECRYPT_METHOD := 'DECRYPTED;VERIFIED'.
- TARGET_NAME := File name of the message extracted from the PGP document. 'NOT_SET' if no file name exists.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acDecrypt fault.
Message is no PGP document	Error. Throws acDecrypt fault.
Decryption key is not matching	Error. Throws acDecrypt fault.
Passphrase is wrong	Error. Throws acDecrypt fault.
Verification key is not matching	Error. Throws acDecrypt fault.

Policy:

ALLOWANCE/PAYLOAD-DECRYPT

Method: Verify with key and decrypt with passphrase

This method verifies and decrypts the current process attachment. Uses the *DecryptVerify* operation of the *OpenPGP Module* component (refer to chapter *10.3.4 DecryptVerify* of the manual) for more details).

Field name	Value	Description
Password	*<Password>*[1]	?????? Optional. Editable password field. The password required by decryption. Properties, system variables and XPath expressions are not supported
Verify alias	*<Ref>*[2]	Optional. Non-editable text field with Keystore Manager selection dialog. The PGP public key required by verification.

Field name	Value	Description
Abort on error	True/False	Refer to Common Fields.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Secure password.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- Current process attachment: PGP document.

Result:

- New process attachment: decrypted message. If detached signature has been verified, the attachment contains the message loaded from *detachedSignature attachmentRef.*

Sets the following properties:

- DECRYPT_METHOD := 'DECRYPTED;VERIFIED'.
- TARGET_NAME := File name of the message extracted from the PGP document. 'NOT_SET' if no file name exists.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acDecrypt fault.
Message is no PGP document	Error. Throws acDecrypt fault.
Password is wrong	Error. Throws acDecrypt fault.
Verification key is not matching	Error. Throws acDecrypt fault.

Policy:

ALLOWANCE/PAYLOAD-DECRYPT

Method: Decrypt with key and no verify (auto)

This method decrypts the current process attachment with a PGP private key from the keystore manager. If the current process attachment is signed, this signature is ignored. Uses the *AutoDecode* operation of the *OpenPGP Module* component (refer to chapter *10.3.5 AutoDecode (link)*of the *OpenPGP Module* manual for more details).

Field name	Value	Description
Decrypt alias	*<Ref>*[1]	Optional. Non-editable text field with Keystore Manager selection dialog. The PGP private key required by the decryption.
Key passphrase (decrypt)	*<Passphrase>*[2]	Optional. Editable password field. The password required by decryption. Properties, system variables and XPath expressions not supported.
Abort on error	True/False	Refer to Common Fields.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Secure passphrase.
3) Arbitrary text.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- Current process attachment: PGP document.

Result:

- New process attachment: decrypted message.

Sets the following properties:

- DECRYPT_METHOD := 'DECRYPTED'.
- TARGET_NAME := File name of the message extracted from the PGP document. 'NOT_SET' if no file name exists.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acDecrypt fault.
Message is no PGP document	Error. Throws acDecrypt fault.
Decryption key is not matching	Error. Throws acDecrypt fault.
Passphrase is wrong	Error. Throws acDecrypt fault.

Policy:

ALLOWANCE/PAYLOAD-DECRYPT

1.11.6 acPGPEncryptMD

Caption: PGPEncryptMD.

Throws Fault=acPGPEncryptMD.

The *acPGPEncryptMD* activity is a single file activity that signs and/or encrypts the current process attachment. Additional features include ASCII armoring, compressing, and changing the destination queue for 'big' files. Uses the *mdCompose* operation of the *OpenPGP Module* component (refer to chapter *10.1.1 mdCompose* of the *OpenPGP Module* manual for more details).

The *acPGPEncryptMD* activity supports the new PGP master data nodes of BIS 6.5.2 (delivered with hotfix B6-652-095). Any usage on an older version will fail.

Field name	Value	Description
PGP own adress	*<Ref>*[1]	Mandatory. Editable selection field. Master data OpenPGP personality specifiying own signature keys and/or encryption keys. The *PGP own address* supports: • System variables (page 392), e.g. '${PGP_OWN}'. • Properties (page 392), e.g. '$[PGP_OWN]'. • XPath expressions (page 392), e.g. $(upper-case('pgppartner')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
PGP partner adress	*<Ref>*[1]	Mandatory. Editable selection field. Master data OpenPGP partner specifying ASCII armor, compression, signature, and encryption settings. The *PGP partner address* supports: • System variables (page 392), e.g. '${PGP_PARTNER}'. • Properties (page 392), e.g. '$[PGP_PARTNER]'. • XPath expressions (page 392), e.g. $(upper-case('pgppartner')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Target filename	*<Filename>*[2]	Optional. Editable text field. File name to use for the signed and/or encrypted and/or compressed document. If empty, the attachment ID is used. The *Target filename* supports: • System variables (page 392), e.g. '${FILENAME}'. • Properties (page 392), e.g. '$[FILENAME]'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Pay attention when entering file names. Because of XPath expression support it is possible to enter invalid file names such as file names consisting, for instance, of asterisks or blanks. If the file name consists of 1 or more blanks only, validating the signed file with the *acPGPDecryptMD* (page 179) activitiy results in a TARGET_NAME property with the value 'NOT_SET'.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Valid file name.
3) Arbitrary text.

Additional configuration:

System variable	Description
MFT-PGP-THRESHOLD-SIZE	Default value 100 MB (100.000.000B). Threshold for file size. Needed for special handling of 'big' files.
MFT-PGP-BIG-FILE-DESTINATION	Default value 'sync:'. Destination queue for signing and/or encrypting if the current process attachment is bigger than the MFT-PGP-THRESHOLD-SIZE system variable. Destination queue for handling smaller PGP documents is always 'sync:'. Possible use case is specification of a destination queue on another node for 'big' files.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- Current process attachment.

Result:

- New process attachment: resulting PGP encoded message.

Sets the following properties:

- ENCRYPT_METHOD := 'SIGNED;ENCRYPTED'.
- TARGET_NAME := The file name for the signed and encrypted document if specified, otherwise 'NOT_SET'.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acPGPEncryptMD fault.
Incorrect master data	Error. Throws acPGPEncryptMD fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP

1.11.7 acPGPDecryptMD

Caption: PGPDecryptMD.

Throws Fault=acPGPDecryptMD.

The *acPGPDecryptMD* activity is a single file activity that verifies the signature and/or decrypts the current process attachment, expecting a PGP encoded document. Uses the *mdDecompose* operation of the *OpenPGP Module* component (refer to chapter *10.3.1 mdDecompose* of the *OpenPGP Module* manual) for more details).

The *acPGPDecryptMD* activity supports the new PGP master data nodes of BIS 6.5.2 (delivered with hotfix B6-652-095). Any usage on an older version will fail.

Field name	Value	Description
PGP own adress	*<Ref>*[1]	Mandatory. Default value '<detect automatically>'. Editable selection field. Master data OpenPGP personality specifiying own signature keys and/or encryption keys. The *PGP own address* supports: • System variables (page 392), e.g. '${PGP_OWN}'. • Properties (page 392), e.g. '$[PGP_OWN]'. • XPath expressions (page 392), e.g. $(upper-case('pgppartner')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
PGP partner adress	*<Ref>*[1]	Mandatory. Default value '<detect automatically>'. Editable selection field. Master data OpenPGP partner specifying ASCII armor, compression, signature, and encryption settings. The *PGP partner address* supports: • System variables (page 392), e.g. '${PGP_PARTNER}'. • Properties (page 392), e.g. '$[PGP_PARTNER]'. • XPath expressions (page 392), e.g. $(upper-case('pgppartner')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.

Additional configuration:

System variable	Description
MFT-PGP-THRESHOLD-SIZE	Default value 100 MB (100.000.000B). Threshold for file size. Needed for special handling of 'big' files.
MFT-PGP-BIG-FILE-DESTINATION	Default value 'sync:'. Destination queue for signing and/or encrypting if the current process attachment is bigger than the MFT-PGP-THRESHOLD-SIZE system variable. Destination queue for handling smaller PGP documents is always 'sync:'. Possible use case is specification of a destination queue on another node for 'big' files.

Preconditions:

- OpenPGP Module adapter started.
- Java unlimited strength policy enabled.
- Current process attachment: PGP encoded message.

Result:

- New process attachment: resulting verified and/or decrypted message.

Sets the following properties:

- DECRYPT_METHOD := 'DECRYPTED;VERIFIED'.

- TARGET_NAME := File name of the message extracted from the PGP document if specified, otherwise 'NOT_SET'.

Possible cases of failure:

Possible case of failure	Success or error behavior
OpenPGP Module adapter not started	Error. Throws acPGPDecryptMD fault.
Incorrect master data	Error. Throws acPGPDecryptMD fault.

Policy:

ALLOWANCE/PAYLOAD-DECRYPT

1.11.8 acPDFSignature

Caption: PDFSignature.

Throws Fault=acPDFSignature.

The *acPDFSignature* activity is a single file activity that sign and verifies a PDF document stored as current process attachment. Uses the *signPDFDocument* operation and the *verify PDFDocument* operation of the *PDF Signature* component/*PDF Secure* adapter (refer to chapter *7.1 signPDFDocument (link)* and chapter *7.2 verifyPDFDocument (link)* of the *PDF Signature Module* manual for more details).

Field name	Value	Description
Method		Mandatory. Selection field. Distinguishes between signing a PDF document and verifying a signed document.
	Sign	Default. Signs the PDF document. Uses the *signPDFDocument* operation. Makes *Digest algorith* and *Signature visibility in pdf* visible.
	Verify	Verifies the signature of a PDF document. Expects a signed PDF document. Uses the *verify PDFDocument* operation

Method: Sign

This Sign method signs a PDF document attached as current process attachment. Uses the *signPDFDocument* operation of the *PDF Signature* component/*PDF Secure* adapter (refer to chapter *7.1 signPDFDocument (link)* of the *PDF Signature Module* manual for more details).

Field name	Value	Description
Key alias	*<Ref>*[1]	Optional. Non editable text field with key-alias selection dialog. The alias of the private key/certificate from the Keystore Manager. The *Key alias* supports: • System variables (page 392), ${ALIAS_NAME}'. • Properties (page 392), e.g. '$[ALIAS_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). I If *PDF Signature own address* is given the *Key alias* is ignored.

Field name	Value	Description
Digest algorithm		Mandatory. Non editable selection field. Provides a list of supported digest or hash algorithms.
	MD5	Default. Message-digest alorithm 5. Produces a 128-bit hash value.
	RIPEMD160	Race Integrity Primitives Evaluation Message Digest. Produces a 160-bit hash value.
	SHA1	Secure hash algorithm. SHA-1 hash function. Produces a 160-bit hash value.
	SHA128	Secure hash algorithm. SHA-2 hash function. Produces a 128-bit hash value.
	SHA256	Secure hash algorithm. SHA-2 hash function. Produces a 256-bit hash value.
	SHA384	Secure hash algorithm. SHA-2 hash function. Produces a 384-bit hash value.
	SHA512	Secure hash algorithm. SHA-2 hash function. Produces a 512-bit hash value.
PDF Signature own address	*<Ref>*[1]	Optional.Editable selection field. PDFSecure address master data. Provides bidirectional and own addresses. Overwrites *Key alias*. The *PDF Signature own address* supports: • System variables (page 392), ${OWN_ADDRESS}'. • Properties (page 392), e.g. '$[OWN_ADDRESS]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Document password	*<Password>*[2]	Optional. Editable password field. The document owner password. Properties, system variables and XPath expressions are not supported.
Signature visibility in pdf	True	The signature is integrated as signature field into the PDF document. Makes visible the *Background image of signature*, *Page of signature*, *Horizontal upper right position*, and *Vertical upper right position* fields below.
	False	The signature is not integrated into the PDF document.
Background image of signature	*<FilePath>*[3]	Optional. Editable text field. File path and name of background image for the graphic part of the signature. The *Background image of signature* supports: • System variables (page 392), ${IMAGE_PATH}'. • Properties (page 392), e.g. '$[IMAGE_PATH]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Signature visibility in pdf* is switched on.
Page of signature	*<Int>*[4]	Mandatory. Editable number field. Default value 1. The page where the signature field is placed. Visible when *Signature visibility in pdf* is switched on. Properties, system variables and XPath expressions are not supported.

Field name	Value	Description
Horizontal upper right position	*<Int>*[4]	Mandatory. Editable number field Default value 300. Minimum value 1, maximum value 1600. The upper right X coordinate of the signature field. Visible when *Signature visibility in pdf* is switched on. Properties, system variables and XPath expressions are not supported.
Vertical upper right position	*<Int>*[4]	Mandatory. Editable number field Default value 200. Minimum value 1, maximum value 1000. The upper right Y coordinate of the signature field. Visible when *Signature visibility in pdf* is switched on. Properties, system variables and XPath expressions are not supported.
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Logmessage	*<Text>*[5]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Secure password.
3) Valid file path and name.
4) Integer number.
5) Arbitrary text.

Preconditions:

- PDFSecure Module adapter started.
- Available process attachment: PDF document expected.

Result:

- New process attachment: the signed PDF document.

Sets the following properties:

- PDF_VERIFY_RESULT := 'Document verified'.

Possible cases of failure:

Possible case of failure	Success or error behavior
PDFSecure Module adapter not started	Error. Throws acPDFSignature fault.
No key alias and no own address with key alias	Error. Throws acPDFSignature fault.
No private key as key alias	Error. Throws acPDFSignature fault.
Non existing *Background file of signature*	Error. Throws acPDFSignature fault.
Invalid page number	Error. Throws acPDFSignature fault.

Policy: none.

Method: Verify

This Verify method signs a signed PDF document attached as current process attachment. Uses the *verify PDFDocument* operation of the *PDF Signature* component/*PDF Secure* adapter (refer to chapter *7.2 verifyPDFDocument (link)* of the *PDF Signature Module* manual for more details).

I PDF signature verification only supports the 'SHA1' digest algorithm. In case of other digest algorithms, this activity method fails.

Field name	Value	Description
Key alias	*<Ref>*[1]	Optional. Editable text field. The alias of the private key/certificate from the Keystore Manager. The *Key alias* supports: • System variables (page 392), ${ALIAS_NAME}'. • Properties (page 392), e.g. '$[ALIAS_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). I If *PDF Signature partner address* is given the *Key alias* is ignored.
PDF Signature partner address	*<Ref>*[1]	Optional.Editable selection field. PDFSecure address master data. Provides bidirectional and partner addresses. Overwrites *Key alias*. The *PDF Signature partner address* supports: • System variables (page 392), ${PARTNER_ADDRESS}'. • Properties (page 392), e.g. '$[PARTNER_ADDRESS]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Document password	*<Password>*[2]	Optional. Editable password field. The document owner password. Properties, system variables and XPath expressions are not supported.
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Secure password.
3) Arbitrary text.

Preconditions:

- PDFSecure Module adapter started.
- Available process attachment: PDF document expected.

Result:

- New process attachment: the verified PDF document.

Sets the following properties:

- PDF_VERIFY_RESULT := 'Document verified'.

Possible cases of failure:

Possible case of failure	Success or error behavior
PDFSecure Module adapter not started	Error. Throws acPDFSignature fault.
No certificate provided	Error. Throws acPDFSignature fault.
Digest algorithm other than 'SHA1'	Error. Throws acPDFSignature fault.

Policies: none.

1.11.9 acSignXML

Caption: SignXML.

Throws Fault=acSignXML.

The *acSignXML* activity is a single file activity that sign the current process attachment (XML document). Several types of XML digital signatures are provided including enveloped and detached signatures. The digital signature can be applied to a part or totality of the XML document. Uses the signXMLDocument, the createDetachedSignature, and the signXadesEPES operations of the XMLDsig adapter.

Field name	Value	Description
Signature type		Mandatory. Non editable selection field. Provides several XML digital signature types, such as enveloped and detached signatures.
	Create detached signature	Sign the XML document and use a detached signature.
	Create enveloping signature	Creates an enveloping digital signature that contains the signed data.
	Create signature only	Do not sign the XML document. Create the XML digital signature in an attachment of its own.
	Sign XML	Default. Sign the XML document and use an enveloped signature.
	XADES	Sign the XML document with a XADES EPES signature.

Signature type: Sign XML

This signature type uses the enveloped type of XML digital signature. Uses the *signXMLDocument* operation of the *XMLDsig Module* component (refer to chapter *1.44.2 signXMLDocument* of the *BIS6 Base Components* manual) or to chapter *8.1 signXMLDocument* of the *XMLDsig Module* manual *(link)* for more details).

Field name	Value	Description
Use masterdata	True	Default. Uses certificates from XMLDsig masterdata. Makes *XML DSIG own address address* visible.
	False	Offers direct certificate selection from the Keystore Manager. Makes *Key alias* visible.
XML DSIG own address	*<Ref>*[1]	Mandatory. Non editable selection field. Master data XMLDsig own address or bidirectional address providing the own signature certificate(s). Visible when *Use masterdata* is switched on.

Field name	Value	Description
Key alias	*<Ref>*[1]	Mandatory. Non editable field with key-alias selection dialog. Alias of the private key/certificate. Visible when *Use masterdata* is switched off.
Digest algorithm		Mandatory. Non editable selection field. Provides a list of available digest or hash algorithms.
	MD5	Message-digest alorithm 5. Produces a 128-bit hash value.
	RIPEMD160	Race Integrity Primitives Evaluation Message Digest. Produces a 160-bit hash value.
	SHA1	Default. Secure hash algorithm. SHA-1 hash function. Produces a 160-bit hash value.
	SHA256	Secure hash algorithm. SHA-2 hash function. Produces a 256-bit hash value.
	SHA384	Secure hash algorithm. SHA-2 hash function. Produces a 384-bit hash value.
	SHA512	Secure hash algorithm. SHA-2 hash function. Produces a 512-bit hash value.
Canonicalization algorithm		Mandatory. Non editable selection field. Provides a list of XML canonicalization algorithms that are used for converting the XML document into the canonical form.
	C14N11-Omit-comments	C14N11. Canonical XML version 1.1 (W3C). Does not include the serialization of XML comments.
	C14N11-With-comments	C14N11. Canonical XML version 1.1 (W3C). Includes the serialization of XML comments.
	C14N-Excl-Omit-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Does not include the serialization of XML comments.
	C14N-Excl-With-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Includes the serialization of XML comments.
	Omit-comments	Default. Does not include the serialization of XML comments.
	With-comments	Includes the serialization of XML comments.
Canonicalization method		Mandatory. Non editable selection field. Provides a list of canonicalization algorithms to be applied to the *SignedInfo* element before the signature calculations.
	C14N11-Omit-comments	C14N11. Canonical XML version 1.1 (W3C). Does not include the serialization of XML comments.
	C14N11-With-comments	C14N11. Canonical XML version 1.1 (W3C). Includes the serialization of XML comments.
	C14N-Excl-Omit-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Does not include the serialization of XML comments.
	C14N-Excl-With-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Includes the serialization of XML comments.
	Omit-comments	Default. Does not include the serialization of XML comments.
	With-comments	Includes the serialization of XML comments.
Root element	*<Name>*[2]	Optional. Editable text field. Case sensitive XML element name. The element in the XML document that is to be signed. If it is empty the complete XML document is signed.

Field name	Value	Description
		The *Root element* supports: • System variables (page 392), e.g. '${ROOT_ELEMENT}'. • Properties (page 392), e.g. '$[ROOT_ELEMENT]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Add public key to signature	True	Default. Add the public key to the structure of the signature element.
	False	Do not add a public key.
Add whole certificate to signature	True	Default. Add the certificate to the signature element.
	False	Do not add a certificate.
Skip signed output normatlization	True	Default. Do not canonicalize the output of the signed XML document.
	False	Canonicalize the output of signed XML document.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary XML tag name.
3) Arbitrary text.

Preconditions:

- XMLDsig Module adapter started.
- Current process attachment: XML document.

Result:

- New process attachment: signed XML document.

Sets the following properties:

- SIGN_RESULT := 'SUCCESS'.
- SIGN_METHOD := 'SIGN' (signature type).

Possible cases of failure:

Possible case of failure	Use masterdata	Success or error behavior
Invalid file format	-	Error. Throws acSignXML fault.
XML DSIG own address without signature key	On	Error. Throws acSignXML fault.
XML DSIG partner address without authentication key	On	Success. XML document is signed.
Non private key as key alias	Off	Error. Throws acSignXML fault.

Policy: none.

Signature type: Create detached signature

This signature type uses the detached type of XML digital signature. Uses the *createDetachedSignature* operation of the *XMLDsig Module* component (refer to chapter *1.44.1 createDetachedSignature* of the *BIS6 Base Components* manual) or to chapter *8.3 createDetachedSignature* of the *XMLDsig Module* manual *(link)* for more details).

Field name	Value	Description
Use masterdata	True	Default. Uses certificates from XMLDsig masterdata. Makes *XML DSIG own address* visible.
	False	Offers direct certificate selection from the Keystore Manager. Makes *Key alias* visible.
XML DSIG own address	*<Ref>*[1]	Mandatory. Non editable selection field. Master data XMLDsig own address or bidirectional address providing the own signature certificate(s). Visible when *Use masterdata* is switched on.
Key alias	*<Ref>*[1]	Mandatory. Non editable field with key-alias selection dialog. Alias of the private key/certificate. Visible when *Use masterdata* is switched off.
Digest algorithm		Mandatory. Non editable selection field. List of available digest or hash algorithms.
	MD5	Message-digest alorithm 5. Produces a 128-bit hash value.
	RIPEMD160	Race Integrity Primitives Evaluation Message Digest. Produces a 160-bit hash value.
	SHA1	Default. Secure hash algorithm. SHA-1 hash function. Produces a 160-bit hash value.
	SHA256	Secure hash algorithm. SHA-2 hash function. Produces a 256-bit hash value.
	SHA384	Secure hash algorithm. SHA-2 hash function. Produces a 384-bit hash value.
	SHA512	Secure hash algorithm. SHA-2 hash function. Produces a 512-bit hash value.
Canonicalization algorithm		Mandatory. Non editable selection field. Provides a list of canonicalization algorithms that are used for converting the XML document into the canonical form.
	C14N11-Omit-comments	C14N11. Canonical XML version 1.1 (W3C). Does not include the serialization of XML comments.
	C14N11-With-comments	C14N11. Canonical XML version 1.1 (W3C). Includes the serialization of XML comments.
	C14N-Excl-Omit-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Does not include the serialization of XML comments.
	C14N-Excl-With-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Includes the serialization of XML comments.
	Omit-comments	Default. Does not include the serialization of XML comments.
	With-comments	Includes the serialization of XML comments.
Canonicalization method		Mandatory. Non editable selection field. Provides a list of canonicalization algorithms to be applied to the *SignedInfo* element before the signature calculations.
	C14N11-Omit-comments	C14N11. Canonical XML version 1.1 (W3C). Does not include the serialization of XML comments.

Field name	Value	Description
	C14N11-With-comments	C14N11. Canonical XML version 1.1 (W3C). Includes the serialization of XML comments.
	C14N-Excl-Omit-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Does not include the serialization of XML comments.
	C14N-Excl-With-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Includes the serialization of XML comments.
	Omit-comments	Default. Does not include the serialization of XML comments.
	With-comments	Includes the serialization of XML comments.
Root element	*<Name>*[2]	Optional. Editable text field. Case sensitive XML element name. The root element in the XML document where the detached signature is to be located. If it is empty the detached signature element is located as last child of the root element. The *Root element* supports: • System variables (page 392), e.g. '${ROOT_ELEMENT}'. • Properties (page 392), e.g. '$[ROOT_ELEMENT]'. XPath expressions are not supported.
Element to sign		Mandatory. Editable text field. Default value 'id-?'. The ID attribute of the element in the XML document that is to be signed. The *Element to sign* supports: • System variables (page 392), e.g. '${ELEMENT_TO_SIGN}'. • Properties (page 392), e.g. '$[ELEMENT_TO_SIGN]'.
Add public key to signature	True	Default. Add the public key to the structure of the signature element.
	False	Do not add a public key.
Add whole certificate to signature	True	Default. Add the certificate to the signature element.
	False	Do not add a certificate.
Add WSSE security element	True	Default. Creates and adds extra WSSE elements to the <KeyInfo> element. ⚠ This flag is **deprecated** and shall not be used anymore. It is maintained for backward compatibility reasons only.
	False	Create standard <KeyInfo> element.
Skip signed output normalization	True	Default. Do not canonicalize the output of the signed XML document.
	False	Canonicalize the output of signed XML document.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary XML tag name.
3) Arbitrary text.

Preconditions:

- XMLDsig Module adapter started.
- Current process attachment: XML document.

Result:

- New process attachment: signed XML document.

Sets the following properties:

- SIGN_RESULT := 'SUCCESS'.
- SIGN_METHOD := 'DETACHED' (signature type).

Possible cases of failure:

Possible case of failure	Use masterdata	Success or error behavior
Invalid file format	-	Error. Throws acSignXML fault.
Invalid or unresolvable element to sign	-	Error. Throws acSignXML fault.
XML DSIG own address without signature key	On	Error. Throws acSignXML fault.
XML DSIG partner address without authentication key	On	Error. Throws acSignXML fault.
Non private key as key alias	Off	Error. Throws acSignXML fault.

Policy: none.

Signature type: Create enveloping signature

This signature type uses the enveloping type of XML digital signature. Uses the *createDetachedSignature* operation of the *XMLDsig Module* component (refer to chapter *8.7 createDetachedSignature* of the *XMLDsig Module* manual *(link)* for more details).

Needs at least BIS6.5.2 SP46. XMLDsig adapter versions of earlier versions do not support this operation

Field name	Value	Description
Use masterdata	True	Default. Uses certificates from XMLDsig masterdata. Makes *XML DSIG own address* visible.

Field name	Value	Description
	False	Offers direct certificate selection from the Keystore Manager. Makes *Key alias* visible.
XML DSIG own address	*<Ref>*[1]	Mandatory. Non editable selection field. Master data XMLDsig own address or bidirectional address providing the own signature certificate(s). Visible when *Use masterdata* is switched on.
Key alias	*<Ref>*[1]	Mandatory. Non editable field with key-alias selection dialog. Alias of the private key/certificate. Visible when *Use masterdata* is switched off.
Digest algorithm		Mandatory. Non editable selection field. List of available digest or hash algorithms.
	MD5	Message-digest alorithm 5. Produces a 128-bit hash value.
	RIPEMD160	Race Integrity Primitives Evaluation Message Digest. Produces a 160-bit hash value.
	SHA1	Default. Secure hash algorithm. SHA-1 hash function. Produces a 160-bit hash value.
	SHA256	Secure hash algorithm. SHA-2 hash function. Produces a 256-bit hash value.
	SHA384	Secure hash algorithm. SHA-2 hash function. Produces a 384-bit hash value.
	SHA512	Secure hash algorithm. SHA-2 hash function. Produces a 512-bit hash value.
Canonicalization algorithm		Mandatory. Non editable selection field. Provides a list of canonicalization algorithms that are used for converting the XML document into the canonical form.
	C14N11-Omit-comments	C14N11. Canonical XML version 1.1 (W3C). Does not include the serialization of XML comments.
	C14N11-With-comments	C14N11. Canonical XML version 1.1 (W3C). Includes the serialization of XML comments.
	C14N-Excl-Omit-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Does not include the serialization of XML comments.

Field name	Value	Description
	C14N-Excl-With-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Includes the serialization of XML comments.
	Omit-comments	Default. Does not include the serialization of XML comments.
	With-comments	Includes the serialization of XML comments.
Canonicalization method		Mandatory. Non editable selection field. Provides a list of canonicalization algorithms to be applied to the *SignedInfo* element before the signature calculations.
	C14N11-Omit-comments	C14N11. Canonical XML version 1.1 (W3C). Does not include the serialization of XML comments.
	C14N11-With-comments	C14N11. Canonical XML version 1.1 (W3C). Includes the serialization of XML comments.
	C14N-Excl-Omit-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Does not include the serialization of XML comments.
	C14N-Excl-With-comments	C14N. Exclusive XML canonicalization version 1.0 (W3C). Includes the serialization of XML comments.
	Omit-comments	Default. Does not include the serialization of XML comments.

Field name	Value	Description
	With-comments	Includes the serialization of XML comments.
Root element	*<Name>*[2]	Optional. Editable text field. Case sensitive XML element name. The root element in the XML document where the detached signature is to be located. If it is empty the detached signature element is located as last child of the root element. The *Root element* supports: • System variables (page 392), e.g. '${ROOT_ELEMENT}'. • Properties (page 392), e.g. '$[ROOT_ELEMENT]'. XPath expressions are not supported.
Element to sign		Mandatory. Editable text field. Default value 'id-?'. The ID attribute of the element in the XML document that is to be signed. The *Element to sign* supports: • System variables (page 392), e.g. '${ELEMENT_TO_SIGN}'. • Properties (page 392), e.g. '$[ELEMENT_TO_SIGN]'.
Add public key to signature	True	Default. Add the public key to the structure of the signature element.
	False	Do not add a public key.
Add reference to signature	True	Default. Aa reference of the KeyInfo element containing the X509Data (public key, certificate) will be added to the SignedInfo element.
	False	Do not add a reference.
Add xades signed properties	True	The xades:SignedProperties element containing xades:SigningTime will be appended to the structure of the signature element and additionally its reference will be added to the SignedInfo element.
	False	Default.
Remove XML declaration	True	Removes XML declarations from the XML file.

Field name	Value	Description
	False	Default.
Xades namespace		Mandatory. Non-editable selection field. Used only when *Add xades signed properties* is switched on. Provides supported XAdES namespace versions that will be used in the generated signature element.
	http://uri.etsi.org/01903/v1.2.2#	Version 1.2.2 (2004)
	http://uri.etsi.org/01903/v1.3.2#	Version 1.3.2 (2006)
	http://uri.etsi.org/01903/v1.4.1#	Version 1.4.1 (2009)
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary XML tag name.
3) Arbitrary text.

Preconditions:

- XMLDsig Module adapter started.
- Current process attachment: XML document.

Result:

- New process attachment: signed XML document.

Sets the following properties:

- SIGN_RESULT := 'SUCCESS'.
- SIGN_METHOD := 'DETACHED' (signature type).

Possible cases of failure:

Possible case of failure	Use masterdata	Success or error behavior
Invalid file format	-	Error. Throws acSignXML fault.
Invalid or unresolvable element to sign	-	Error. Throws acSignXML fault.
XML DSIG own address without signature key	On	Error. Throws acSignXML fault.
XML DSIG partner address without authentication key	On	Error. Throws acSignXML fault.
Non private key as key alias	Off	Error. Throws acSignXML fault.

Policy: none.

Signature type: Create signature only

This signature type creates the signature for the current process attachment, actually the encrypted digest of the XML document, and returns it as new attachment. This new process attachment can be bookmarked with the *scBookmark (page 6)* activity and used within the *acVerifyXML* (page 198) activity by the 'Verify xml against digest value' signature type in order to verify signed XML documents.

Field name	**Value**	**Description**
Use masterdata	True	Default. Uses certificates from XMLDsig masterdata. Makes *XML DSIG own address* visible.
	False	Offers direct certificate selection from the Keystore Manager. Makes *Key alias* visible.
XML DSIG own address	*<Ref>*[1]	UNUSED! Mandatory. Non editable selection field. Master data XMLDsig own address or bidirectional address providing the own signature certificate(s). Visible when *Use masterdata* is switched on. ⚠ The masterdata mode is currently not supported. The activity fails always. Please use the non-masterdata mode with direct key/alias selection from the Keystore Manager.
Key alias	*<Ref>*[1]	Mandatory. Non editable field with key-alias selection dialog. Alias of the private key/certificate.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.

Preconditions:

- ScriptExecutor adapter started.
- Current process attachment: XML document.

Result:

- New process attachment: contains the created XML signature, i.e. the encrypted digest of the XML document. SHA-1 hash function used.

Sets the following properties:

- SIGN_RESULT := 'SUCCESS'.
- SIGN_METHOD := 'DIGEST' (signature type).

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid file format	Error. Throws acSignXML fault.
Public key as key alias	Error. Throws acSignXML fault.
Masterdata mode switched on	Error. Throws acSignXML fault. ⚠ The masterdata mode is currently not supported. The activity always fails. Please use the non-masterdata mode with direct key/alias selection from the Keystore Manager.

Policy: none.

Signature type: Xades

This signature type uses the enveloped type of XML digital signature. Uses the *signXadesEPES* operation of the *XMLDsig Module* component (refer to chapter *1.44.3 signXadesEPES* of the *BIS6 Base Components* manual or to chapter *8.5 signXadesEPES* of the *XMLDsig Module* manual *(link)* for more details).

Field name	Value	Description
Use masterdata	True	Default. Uses certificates from XMLDsig masterdata. Makes *XML DSIG own address ad dress* visible.
	False	Offers direct certificate selection from the Keystore Manager. Makes *Key alias* visible.
XML DSIG own address	*<Ref>*[1]	Mandatory. Non editable selection field. Master data XMLDsig own address or bidirectional address providing the own signature certificate(s). Visible when *Use masterdata* is switched on.
Key alias	*<Ref>*[1]	Mandatory. Non editable field with key-alias selection dialog. Alias of the private key/certificate. Visible when *Use masterdata* is switched off.
Digest algorithm		Mandatory. Non editable selection field. Provides a list of available digest or hash algorithms.
	MD5	Message-digest alorithm 5. Produces a 128-bit hash value.
	RIPEMD160	Race Integrity Primitives Evaluation Message Digest. Produces a 160-bit hash value.
	SHA1	Default. Secure hash algorithm. SHA-1 hash function. Produces a 160-bit hash value.
	SHA256	Secure hash algorithm. SHA-2 hash function. Produces a 256-bit hash value.

Field name	Value	Description
	SHA384	Secure hash algorithm. SHA-2 hash function. Produces a 384-bit hash value.
	SHA512	Secure hash algorithm. SHA-2 hash function. Produces a 512-bit hash value.
Root element	*<Name>*[2]	Optional. Editable text field. Case sensitive XML element name. The element in the XML document that is to be signed. If it is empty the complete XML document is signed. The *Root element* supports: • System variables (page 392), e.g. '${ROOT_ELEMENT}'. • Properties (page 392), e.g. '$[ROOT_ELEMENT]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')).
Add public key to signature	True	Default. Add the public key to the structure of the signature element.
	False	Do not add a public key.
Add reference to signature	True	Default. Adds a *Reference* element pointing to the *KeyInfo* element to the structure of the signature *SignedInfo* element. This means that the contents of the *KeyInfo* is signed.
	False	The content of the *KeyInfo* is not signed.
Signer role		The role claimed by the signer.
	customer	Customer.
	supplier	Supplier.
	third-party	Third party.
Policy implied	True	Determines the structure of XADES standard element *SignaturePolicyIdentifier* to be composed.
	False	Default.
Skip signed output normalization	True	Default. Do not canonicalize the output of the signed XML document.

Field name	Value	Description
	False	Canonicalize the output of signed XML document.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary XML tag name.
3) Arbitrary text.

Preconditions:

- XMLDsig Module adapter started.
- Current process attachment: XML document.

Result:

- Process attachment: is not changed.

Sets the following properties:

- SIGN_RESULT := 'SUCCESS'.
- SIGN_METHOD := 'XADES' (signature type).

Possible cases of failure:

Possible case of failure	Use masterdata	Success or error behavior
Invalid file format	-	Error. Throws acSignXML fault.
Invalid or unresolvable element to sign	-	Error. Throws acSignXML fault.
XML DSIG own address without signature key	On	Error. Throws acSignXML fault.
XML DSIG partner address without authentication key	On	Error. Throws acSignXML fault.
Non private key as key alias	Off	Error. Throws acSignXML fault.

Policy: none.

1.11.10 acVerifyXML

Caption: VerifyXML.

Throws Fault=acVerifyXML.

The *acVerifyXML* activity is a single file activity that verifies the current process attachment (signed XML document). Uses the *verifyXMLDocument* operation, and the *verifyDetachedSignature* operation of the *XMLDsig adapter*.

Field name	Value	Description
Signature type		Mandatory. Non-editable selection field. Provides several XML digital signature types, such as enveloped and detached signatures.
	VerifyXML	Default. Verify the XML document, expect an enveloped signature.
	Verify xml against digest value	Verify the XML document, expect a stand-alone digest value.
	Verify xml with detached signature	Verify the XML document, expect a detached signature.
Use masterdata	True	Default. Use master data partner address. Makes *XML DSIG partner address* visible.
	False	Do not use master data addresses. Makes *Key alias* visible.

Signature type: VerifyXML
This type verifies the current process attachment (signed XML document expected). Uses the *verifyXMLDocument* operation of the *XMLDsig Module* component (refer to chapter *1.44.4 verifyXMLDocument* of the *BIS6 Base Components* manual or to chapter *8.2 verifyXMLDocument* of the *XMLDsig Module* manual *(link)* for more details).

Field name	Value	Description
XML DSIG partner address	*<Ref>*[1]	Mandatory. Non-editable selection field. Master data XMLDsig partner address or bidirectional address providing the partner authentification certificate(s). Visible when *Use master* data is switched on.
Key alias	*<Ref>*[1]	Mandatory. Non-editable field with Keystore Manager certificate selection dialog. The alias of the public key/certificate from the *Keystore Manager*. Visible when *Use master* data is switched off.
Use key info tag	True	Use the *KeyInfo* data for the process of the signature verification.
	False	Default. Do not use the *KeyInfo* data for the process of the signature verification.
Remove signature from XML	True	Remove the Signature element (i.e. <ds:Signature>) after verification.
	False	Default. Keep the signature element after verification.
Ignore missing signature element	True	If the Signature element (i.e. <ds:Signature>) is not found do not throw an error and simply return the XML document.
	False	Default. Throw an error if the Signature element is not found.
Do OCSP check for embedded cert	True	If the *Use key info tag* is switched on, the certificate embedded in the current XML document is validated against an OCSP provider.
	False	Default. Do not validate against OCSP provider.
Do CRL check for embedded cert	True	If the *Use key info tag* is switched on, the certificate embedded in the current XML document is validated using a Certificate Revocations List (CRL).
	False	Default. Do not validate against CRL.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.

Preconditions:

- XMLDsig Module adapter started.
- Current process attachment: signed XML document.

Result:

- New process attachment: resulting XML document.

Sets the following properties:

- VERIFY_RESULT := 'SUCCESS'.
- VERIFY_METHOD := 'VERIFY'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Master data address without certificates	Error. Throws acVerifyXML fault.
Wrong file format	Error. Throws acVerifyXML fault.
XML document without Signature element	Error. Throws acVerifyXML fault.

Policy: none.

Signature type: Verify xml against digest value
This type verifies the current process attachment (signed XML document expected) comparing it with an existing digest. The necessary digest bookmark can be created by using the activity to create the digest and then by bookmarking this digest with the *scBookmark* activity.

Field name	Value	Description
Key alias	*<Ref>*[1]	Mandatory. Non-editable field with Keystore Manager certificate selection dialog. Visible when *Use master* data is switched off.
Signature from bookmark	*<Name>*[2]	Mandatory. Editable text field. A bookmark reference to an existing digest/signature. The *Signature from bookmark* supports: • Properties (page 392), e.g. '$[MY_BOOKMARK_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). System variables are not supported.
XML DSIG partner address	*<Ref>*[1]	Mandatory. Non-editable selection field. Master data XMLDsig partner address or bidirectional address providing the partner authentification certificate(s). Visible when *Use master* data is switched on.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary name except invalid characters: ", &, <.
3) Arbitrary text.

Preconditions:

- ScriptExecutor adapter started.
- Current process attachment: signed XML document.

Result:

- New process attachment: resulting XML document.

Sets the following properties:

- VERIFY_RESULT := 'SUCCESS'.
- VERIFY_METHOD := 'DIGEST'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Non existing bookmark	Error. Throws acVerifyXML fault.
Wrong file format	Error. Throws acVerifyXML fault.

Policy: none.

Signature type: Verify xml with detached signature
This type verifies the current process attachment (XML document with detached signature expected). Uses the *verifyDetachedSignature* operation of the *XMLDsig Module* component (refer to chapter *1.44.5 verifyDetachedSignature* of the *BIS6 Base Components* manual or to chapter *8.4 verifyDetachedSignature* of the *XMLDsig Module* manual *(link)* for more details).

Field name	Value	Description
XML DSIG partner address	*<Ref>*[1]	Mandatory. Non-editable selection field. Master data XMLDsig partner address or bidirectional address providing the partner authentification certificate(s). Visible when *Use master* data is switched on.
Key alias	*<Ref>*[1]	Mandatory. Non-editable field with Keystore Manager certificate selection dialog. Visible when *Use master* data is switched off.
Use WSSE security element	True	Add extra W*sse elements* while creating the detached signature. ⚠ This flag is **deprecated** and shall not be used anymore. It is maintained for backward compatibility reasons only.
	False	Default. Do not add any W*sse elements*.
Use key info tag	True	Use the *KeyInfo* data for the process of the signature verification.
	False	Default. Do not use the *KeyInfo* data for the process of the signature verification.
Remove signature from XML	True	Remove the Signature element (i.e. <ds:Signature>) after verification.
	False	Default. Keep the signature element after verification.
Ignore missing signature element	True	If the Signature element (i.e. <ds:Signature>) is not found do not throw an error and simply return the XML document.
	False	Default. Throw an error if the Signature element is not found.

Field name	Value	Description
Do OCSP check for embedded cert	True	If the *Use key info tag* is switched on, the certificate embedded in the current XML document is validated against an OCSP provider.
	False	Default. Do not validate against OCSP provider.
Do CRL check for embedded cert	True	If the *Use key info tag* is switched on, the certificate embedded in the current XML document is validated using a Certificate Revocations List (CRL).
	False	Default. Do not validate against CRL.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.

Preconditions:

- XMLDsig Module adapter started.
- Current process attachment: signed XML document.

Result:

- New process attachment: resulting XML document.

Sets the following properties:

- VERIFY_RESULT := 'SUCCESS'.
- VERIFY_METHOD := 'DETACHED'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Master data address without certificates	Error. Throws acVerifyXML fault.
Wrong file format	Error. Throws acVerifyXML fault.
Unsigned XML document	Error. Throws acVerifyXML fault.
XML document with enveloped signature	Success.

Policy: none.

1.12 FTP Activities

This section describes activities that allow remote file handling on an FTP server.

1.12.1 acFTPList

Caption: FTPList.

Throws Fault=acFTPList.

The *acFTPList* activity lists multiple files of a directory on an FTP server and returns a FILELIST object (CSV file) that can be used as direct input to the *scForEach* (page 10) activity providing a loop element. Uses the *listBox* operation from the *FTP Client Adapter* component (refer to chapter *1.13.5 listBox (List Message s)* of the *BIS6 Base Components* manual or to chapter *8.5 listBox (List Messages)* of the *FTP Client Adapter* manual) for more details).

Field name	**Value**	**Description**
File mask	*<Pattern>*[1]	Mandatory. Editable text field. Default value is '*.*'. Case sensitive file list filter acting as include pattern. Processes only files that match this condition. Actually a download mask for the FTP server (refer to **HostDir Syntax** below for more details about the syntax). Overwrites the *Mask* field from the FTP *Address*. The *File mask* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. • Relative paths where the relative path is related to the *Connection* and the *Address* object. The path part does not support any mask patterns. A relative path from *Connection* is overwritten. This means if all relative paths are entered (base directory <BaseDir> in FTP client *Connection*, relative path <RelPathAddr> in FTP client *Address* , and relative path <RelPathAct> in this activity) the resulting file path is the following: <BaseDir>/<RelPathAct>. • System variables (page 392), e.g. '${MY_INCLUDE_PATTERN}'. • Properties (page 392), e.g. '$[MY_INCLUDE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Character sets and character ranges are not supported.
Exclude pattern	*<Pattern>*[2]	Optional. Editable text field. Empty by default. Case sensitive exclusion file mask. Processes only files that do not match this condition. Actually a file list filter used by the BIC mapping that creates the FILELIST object. The *Exclude Mask* field from the FTP *Address* is never used. The *Exclude pattern* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. • Character sets matching exactly one character of the specified set, e.g. [abc]. • Character ranges matching exactly one character of the specified range, e.g. [a-z]. • System variables (page 392), e.g. '${MY_EXCLUDE_PATTERN}'. • Properties (page 392), e.g. '$[MY_EXCLUDE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Relative paths are not supported.
Mapping parameter		Optional. Editable text field. Empty by default.

Field name	Value	Description
		Currently not used.
Address	*<Ref>*[3]	Mandatory. Not editable selection field. Master data FTP client address. Represents an individual FTP user with access to the FTP server specified in *Connection*.
Connection	*<Ref>*[3]	Mandatory. Not editable selection field. Master data FTP client connection. Represents an FTP server and its communication settings.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[3]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Case sensitive FTP file download mask pattern allowing wildcards.
2) Case sensitive file mask pattern allowing wildcards.
3) Reference to master data.
4) Arbitrary text.

Additional configuration:

System variable	Description
MFT-FILELISTCREATOR	Name of mapping for conversion of file list into FILELIST CSV format. ⚠ Changing this mapping affects all list activities with FILELIST format!

Preconditions:

- FTP client adapter started.
- BIC client adapter started.
- Mapping from ${MFT-FILELISTCREATOR} system variable available in repository.

Result:

- New attachment: CSV file containing the file list. Empty CSV file for an empty file list.

Sets the following properties:

- LIST_COUNT := Number of entries in the returned file list.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Success. Returns a list with 0 files.
Directory without permission	Success. Returns a list with 0 files.
Empty directory	Success. Returns a list with 0 files.
File without permission	Success. Returns a list with all files.
Locked file	Success. Returns a list with all files.
Invalid FTP client connection: wrong host or port	Error. Throws acFTPList fault.
Invalid FTP client address: wrong user or password	Error. Throws acFTPList fault.
Invalid FTP client address: wrong relative path	Error. Throws acFTPList fault.

Policy: none.

1.12.2 HostDir Syntax

The SEEBURGER FTP Adapter interprets *HostDir* as the concatenation of *baseDir*, *relativePath,* and *mask*, which are contained in both the *FTP Connection Settings* and *DtFtpRequestType*.

Note: Since the *HostDir* is used as a FTP command argument, it is strongly recommended that *HostDir* is a string formed by any of the 128 ASCII characters except <*CR*> and <*LF*> as stated in RFC959. However, there are FTP servers, which implement the internationalization draft (RFC2640). Such servers support *HostDir* with the UTF-8 characters. Generally, the FTP Adapter works with these servers, but a successfull operation cannot be warranted.

Attention: The *HostDir* command must correspond to the **FTP server OS path conventions**. The FTP client will not take care of correcting the *HostDir* command, e.g. inserting a "/" or a "\", because not all the existing possibilities can be considered.

The *HostDir* must correspond to the FTP server's OS path conventions. The FTP adapter cannot consider any possibility for the *HostDir* to be formed by the combination of the *server type* and the *FTP server OS*. Thus, the FTP client will not adjust the *HostDir* settings, e.g. by inserting a "/" or a "\".

If the *relativePath* is *send* and the *mask* is *invoice*, the file will be sent as a *sendinvoice* instead of a *send/invoice*. It is the user's responsibility to correct the *HostDir* settings.

The *HostDir* specifies the type and the location of uploaded or downloaded files. The behavior described here is only applicable to the standard FTP server type, and to all other server types unless explicitly stated otherwise. This syntax definitely does not apply for VANs.

The syntax of the *HostDir* must confirm to the server site conventions. A plain directory is assumed, if no file name (mask) is given. E.g. the *HostDir* ends with a path separator character. Wildcards are also supported in the file name path (mask) of the *HostDir*. Wildcards are resolved on the client side (e.g. the FTP server is never questioned to resolve the wildcard, since no common syntax is defined in the FTP specification). Supported symbols are "?" that denotes a single arbitrary character and "*" that denotes multiple arbitrary characters.

Note: Most FTP servers do not allow the host path to start with a slash that denotes the root dir, because the path resolution is assumed to be relative to the user's home folder.

1.12.3 acFTPPickFile

Caption: FTPPickFile.

Throws Fault=acFTPPickFile.

The *acFTPPickFile* activity is a single file activity that creates an attachment from the specified file located on an FTP server. Uses the *getFile* operation from the *FTP Client Adapter* component (refer to chapter *1.13.2 getFile (Get Message)* of the *BIS6 Base Components* manual) or to chapter *8.6 getFile (Get Message)* of the *FTP Client Adapter* manual) for more details).

Field name	**Value**	**Description**
File name	*<FileName>*[1]	Mandatory. Editable text file. Empty by default. The valid name of the source file. The *File name* supports: • Relative paths where the relative path is related to the *Connection* and the *Address* object. The path part does not support any mask patterns. A relative path from *Connection* is overwritten. This means if all relative paths are entered (base directory <BaseDir> in FTP client *Connection*, relative path <RelPathAddr> in FTP client *Address* , and relative path <RelPathAct> in this activity) the resulting file path is the following: <BaseDir>/<RelPathAct>. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Address	*<Ref>*[2]	Mandatory. Not editable selection field. Master data FTP client address. Represents an individual FTP user with access to the FTP server specified in *Connection*.
Connection	*<Ref>*[2]	Mandatory. Not editable selection field. Master data FTP client connection. Represents an FTP server and its communication settings.
Abort on error	True/False	Refer to Common Fields (page 3).
Delete picked file	True	Delete the picked file.
	False	Default. Keep the picked file.
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- FTP client adapter started.

Result:

- New process attachment: Picked file.

Sets the following properties:

- TARGET_NAME := The name of the source file to pick.
- TARGET_PATH := The path of the source file to pick.
- FILE_COUNT := The number of picked files, 1 in case of real success, 0 in case of error when the *Abort on error* flag is not switched on.
- DIRECTION := 'IN'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acFTPPickFile fault.
Empty directory.	Error. Throws acFTPPickFile fault.
Directory without permission	Error. Throws acFTPPickFile fault.
Not existing file	Error. Throws acFTPPickFile fault.
Empty file	Success. File is picked up.
File without permission	Error. Throws acFTPPickFile fault.
Locked file	Error. Throws acFTPPickFile fault.
File name with unsupported characters	Error. Throws acFTPPickFile fault.
Invalid FTP client connection: wrong host or port	Error. Throws acFTPPickFile fault.
Invalid FTP client address: wrong user or password	Error. Throws acFTPPickFile fault.
Invalid FTP client address: wrong relative path	Error. Throws acFTPPickFile fault.

Policy: none.

1.12.4 acFTPPolling

Caption: FTP polling.

Throws Fault=acFTPPolling2.

The *acFTPPolling* activity is a multiple files activity that polls files from a specified location on an FTP server that trigger new MFT-FileSystemEvents (refer to chapter *MFT-FTPEvent* in the *MFT Standard Solution - Adapters* manual for more details). This actually represents a file download. Uses the *polling* operation from the *FTP Client Adapter* component (refer to chapter *1.13.6 polling (Polling Messages)* of the *BIS6 Base Components* manual or to chapter *8.3 polling (Polling Messages)* of the *FTP Client adapter* manual for more details).

There are 2 different polling configurations available, one with and one without FTP master data:

Field name	Value	Description
Poll without MD	True	Enter FTP connection data directly.
	False	Use FTP master data.

Poll with FTP master data

Field name	Value	Description
Poll without MD	False	Use the FTP client master data.
File mask	*<Pattern>*[1]	Mandatory. Editable text field. Empty by default. Case sensitive file list filter acting as include pattern. Processes only files that match this condition. The *File mask* supports:

Field name	Value	Description
		- Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. - Relative paths where the relative path is related to the *Connection* and the *Address* object. The path part does not support any mask patterns. A relative path from *Address* is overwritten. - System variables (page 392), e.g. '${MY_FILE_PATTERN}*.dta'. - Properties (page 392), e.g. '$[MY_FILE_PATTERN]'. Character sets and character ranges are not supported.
Address	*<Ref>*2	Mandatory. Not editable selection field. Master data FTP client address. Represents an individual FTP user with access to the FTP server specified in *Connection*.
Connection	*<Ref>*2	Mandatory. Not editable selection field. Master data FTP client connection. Represents an FTP server and its communication settings.
Delete polled files	True	Delete the polled files.
	False	Keep the polled files.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*2	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*3	Optional. Refer to Common Fields (page 3).

1) Case sensitive file mask pattern allowing wildcards.
2) Reference to master data.
3) Arbitrary text.

Poll without FTP master data

Field name	Value	Description
Poll without MD	True	Directly enter all data. Do not use the FTP client master data.
File mask	*<Pattern>*1	Mandatory. Editable text field. Empty by default. Case sensitive file list filter acting as include pattern. Processes only files that match this condition. The *File mask* supports: - Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. - Relative paths. The path part does not support any mask patterns. - System variables (page 392), e.g. '${MY_FILE_PATTERN}*.dta'. - Properties (page 392), e.g. '$[MY_FILE_PATTERN]'. Character sets and character ranges are not supported.
Host	<Host>	Mandatory. Default value is '127.0.0.1'. The host name or IP address of the FTP server.
Port	*<Int>*2	Mandatory. The default value is *'21'*. The port of the FTP server.
User	<UserName>	Mandatory. Editable text field. Empty by default. The user name for establishing a connection to the FTP server.

Field name	Value	Description
Password	*<Password>*[3]	Mandatory. Editable password field. Empty by default. The password for establishing a connection to the FTP server.
Lock file	*<FileName>*[4]	Optional. Editable text field. Empty by default. Name of lock file.
Server type		Optional. Non-editable selection field.Available server types.
	STANDARD	Default. Standard server type if you are unsure when selecting the adequate server.
	<ServerType>	Server types available for selection: ECODEX, GE_MARK_3000, GE_MARK_III, IBM_IE, MVS, STERLING_COMMERCE.
Connection type		Optional. Non-editable selection field. Provides two different connection modes for the communication with the FTP server specified above. Refer to chapter *9.4* in the *FTP Client adapter* manual for further details.
	ACTIVE	Use the active mode.
	PASSIVE	Default. Use the passive mode, also known as Client Managed FTP.
Keep alive	True	Default. Activates the *Keep-alive* mechanism. Refer to chapter *9.3 Keep-alive feature* in the *FTP Client adapter* manual for further details.
	False	Does not activate the *Keep-alive* mechanism.
Delete polled files	True	Delete the polled files.
	False	Keep the polled files.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[5]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[6]	Optional. Refer to Common Fields (page 3).

1) Case sensitive file mask pattern allowing wildcards.
2) Positive integer value denoting a valid port number.
3) Secure password.
4) Valid file name.
5) Reference to master data.
6) Arbitrary text.

Preconditions:

- FTP client adapter started.

Result:

- New *MFT-FTPEvents* for the polled files.

Sets the following properties:

- FILE_COUNT := The number of polled files.
- FILES_LEFT := The number of files that could not be polled.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Success. FILE_COUNT is set to 0.
Empty directory	Success. FILE_COUNT is set to 0.
Directory without permission	Success. FILE_COUNT is set to 0.
Empty file	Success. FILE_COUNT is increased.
Files with and without permission	Success. File is not polled. FILES_LEFT is increased.
Only files without permission	Error. Throws acFTPPolling fault.
Locked files and files with permission	Success. File is not polled. FILES_LEFT is increased.
Only locked file	Error. Throws acFTPPolling fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/FTP ALLOWANCE/SYSTEM-EXECUTE-SCRIPT

1.12.5 mdFTPPolling

Caption: FTP polling.

Throws Fault=acFTPPolling.

⚠ **Deprecated! Please use acFTPPolling instead!**

The *mdFTPPolling* activity is a multiple files activity that polls files from a specified location on an FTP server that trigger new MFT-FileSystemEvents (refer to chapter *MFT-FTPEvent* in the *MFT Standard Solution - Adapters* manual) for more details). This actually represents a file download. Uses the *polling* operation from the *FTP Client Adapter* component (refer to chapter *1.13.6 polling (Polling Messages)* of the *BIS6 Base Components* manual) or to chapter *8.3 polling (Polling Messages)* of the *FTP Client Adapter* manual for more details).

Field name	Value	Description
File mask	*<Pattern>*[1]	Mandatory. Editable text field. Default value is '*' matching all available files. Case sensitive file list filter acting as include pattern. Processes only files that match this condition. The *File mask* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. • Relative paths where the relative path is related to the *Connection* and the *Address* object. The path part does not support any mask patterns. A relative path from *Address* is overwritten. • System variables (page 392), e.g. '${MY_FILE_PATTERN}*.dta'. • Properties (page 392), e.g. '$[MY_FILE_PATTERN]'. Character sets and character ranges are not supported.
Address	*<Ref>*[2]	Mandatory. Not editable selection field. Master data FTP client address. The address object represents an individual FTP user with access to the FTP server specified in *Connection*.

Field name	Value	Description
Connection	*<Ref>*[2]	Mandatory. Not editable selection field. Master data FTP client connection. The connection object represents an FTP server and its communication settings.
Delete polled files	True	Delete the polled files.
	False	Keep the polled files.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Case sensitive file mask pattern allowing wildcards.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- FTP client adapter started.

Result:

- New *MFT-FTPEvents* for the polled files.

Sets the following properties:

- FILE_COUNT := The number of polled files.
- FILES_LEFT := The number of files that could not be polled.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Success. FILE_COUNT is set to 0.
Empty directory	Success. FILE_COUNT is set to 0.
Directory without permission	Success. FILE_COUNT is set to 0.
Empty file	Success. FILE_COUNT is increased.
Files with and without permission	Success. File is not polled. FILES_LEFT is increased.
Only files without permission	Error. Throws acFTPPolling fault.
Locked files and files with permission	Success. File is not polled. FILES_LEFT is increased.
Only locked file	Error. Throws acFTPPolling fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/FTP ALLOWANCE/SYSTEM-EXECUTE-SCRIPT

1.12.6 acFTPProcess

Caption: FTPProcess.

Throws Fault=acFTPProcess.

The *acFTPProcess* activity represents the FTP interaction with the server. It can upload, download, or both in a single operation. If a file is downloaded it is placed as message attachment in the response without creating a new FTPEvent. Uses the *processFtp* operation from the *FTP Client Adapter* component (refer to chapter *1.13.7 processFtp (FTP Processing)* of the *BIS6 Base Components* manual or to chapter *8.4 processFtp (FTP Processing)* of the *FTP Client Adapter* manual for more details).

Note: The *processFtp* operation has a default script only for a BAAN server type. For the standard FTP server type, the script must be explicitly specified in the *scriptFile*, or *script* field on the request. Refer to chapter *10 Scripting (link)*in the *FTP Client Adapter* manual for further details about scripts.

Script file	*<URI>*[1]	Optional. Editable text field. Empty by default. This is the script file URI where the script for the FTP server is located. If there is no script file available the *Script* needs to be filled. Supports: • Script parameters from *Specific data* field. • System variables (page 392), e.g. '${QA_HOME_DIR}\FTP_Scripts\ftpscript.dta'. • Properties (page 392), e.g. '$[SCRIPT_FILE_URI]'. Wildcards are not supported.
Script	*<FTPScript>*[2]	Optional. Editable text field. Empty by default. This is the script content for the FTP server. This script will be used if no *scriptFile* URI is present. Supports: • Script parameters from *Specific data* field. System variables and properties are not supported.
Specific data	*<KVList>*[3]	Optional. Editable text field. Empty by default. Parameters and variable definitions to be used in the script file, for instance: "HOST=172.0.0.1". Supports: • Maximum number of key-value pairs is 2000. Additional key-value pairs are ignored.
Address	*<Ref>*[4]	Mandatory. Not editable selection field. Master data FTP client address. The address object represents an individual FTP user with access to the FTP server specified in *Connection*.
Connection	*<Ref>*[4]	Mandatory. Not editable selection field. Master data client connection. The connection object represents an FTP server and its communication settings.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[4]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[5]	Optional. Refer to Common Fields (page 3).

1) Valid URI.
2) Valid FTP script.
3) Semicolon separated key-value list.
4) Reference to master data.
5) Arbitrary text.

Preconditions:

- FTP client adapter started.

Result:

- New attachment if the result of the script execution includes an attachment (i.e. ftp download).

Sets no properties.

Possible cases of failure:

Possible case of failure	Success or error behavior
No script available (since fields are optional)	Error. Throws acFTPProcess fault.
Invalid *script file* URI	Error. Throws acFTPProcess fault.
Not existing *script file*	Error. Throws acFTPProcess fault.
Script file without permission	Error. Throws acFTPProcess fault.
Locked *script file*	Error. Throws acFTPProcess fault.
Empty *script file*	Success. No action.
Missing parameters for *script file*	Error. Throws acFTPProcess fault.
Invalid content of *Script file*	Error. Throws acFTPProcess fault.
Script file name with unsupported characters	Error. Throws acFTPProcess fault.
Invalid content of *Script*	Error. Throws acFTPProcess fault.
Invalid format of *Specific data*	Error. Throws acFTPProcess fault.
Invalid FTP client connection: wrong host or port	Error. Throws acFTPProcess fault.
Invalid FTP client address: wrong user	Error. Throws acFTPProcess fault.
Invalid FTP client address: wrong relative path	Error. Throws acFTPProcess fault.

Policy: none.

1.12.7 acFTPRemoveFile

Caption: FTPRemoveFile.

Throws Fault=acFTPRemoveFile.

The acFTPRemoveFile activity is a single file activity that removes a file located on an FTP server. Uses the *pr ocessFtp* operation from the *FTP Client Adapter* component calling it with a file removing script (refer to chapter *1.13.7 processFtp (FTP Processing)* of the *BIS6 Base Components* manual or to chapter *8.4 processFtp (FTP Processing)* of the *FTP Client Adapter* manual for more details).

Field name	Value	Description
Remote file name	<FileName>[1]	Mandatory. Editable text field. Default value ''MyFile.txt'. The valid name of the source file. The *File name* supports: • Relative paths where the relative path is related to the *Connection* and the *Address* object. The path part does not support any mask patterns.

Field name	Value	Description
		A relative path from *Connection* is appended by this relative path if existing. This means if all relative paths are entered (base directory <BaseDir> in FTP client *Connection*, relative path <RelPathAddr> in FTP client *Address*, and relative path <RelPathAct> in this activity) the resulting file path is the following: <BaseDir>/<RelPathAddr>/<RelPathAct>. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Address	*<Ref>*[2]	Mandatory. Non-editable selection field. Master data FTP client address. The address object represents an individual FTP user with access to the FTP server specified in *Connection*.
Connection	*<Ref>*[2]	Mandatory. Non-editable selection field. Master data FTP client connection. The connection object represents an FTP server and its communication settings.
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- FTP client adapter started.

Result:

- Current process attachment: not accessed and not changed.

Sets the following properties:

- TARGET_NAME := The name of the removed file.
- TARGET_PATH := The path of the removed file.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acFTPRemoveFile fault.
Directory without permission	Error. Throws acFTPRemoveFile fault.
Not existing file	Error. Throws acFTPRemoveFile fault.
Empty file	Success. File is picked up.
File without permission	Error. Throws acFTPRemoveFile fault.
Locked file	Error. Throws acFTPRemoveFile fault.
File name with unsupported characters	Error. Throws acFTPRemoveFile fault.

Policy: none.

1.12.8 acFTPRename

Caption: FTPRename.

Throws Fault=acFTPRename.

The acFTPRename activity is a single file activity that renames an existing remote source file to a target remote file on an FTP server. Uses the *processFtp* operation from the *FTP Client Adapter* component calling it with a file renaming script (refer to chapter *1.13.7 processFtp (FTP Processing)* of the *BIS6 Base Components* manual or to chapter *8.4 processFtp (FTP Processing)* of the *FTP Client Adapter* manual for more details).

Field name	Value	Description
Remote file name	*<FileName>*[1]	Mandatory. Editable text field. Empty by default. The name of the target file. Must be a valid file name. The *File name* supports: • Relative paths where the relative path is related to the *Connection* and the *Address* object. The path part does not support any mask patterns. A relative path from *Connection* is appended by this relative path if existing. This means if all relative paths are entered (base directory <BaseDir> in FTP client *Connection*, relative path <RelPathAddr> in FTP client *Address*, and relative path <RelPathAct> in this activity) the resulting file path is the following: <BaseDir>/<RelPathAddr>/<RelPathAct>. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Target file name	*<FileName>*[1]	Mandatory. Editable text field. Empty by default. The name of the target file. Must be a valid file name. The *File name* supports: • Relative paths where the relative path is related to the *Connection* and the *Address* object. The path part does not support any mask patterns. A relative path from *Connection* is appended by this relative path if existing. This means if all relative paths are entered

Field name	Value	Description
		(base directory <BaseDir> in FTP client *Connection*, relative path <RelPathAddr> in FTP client *Address*, and relative path <RelPathAct> in this activity) the resulting file path is the following: <BaseDir>/<RelPathAddr>/<RelPathAct>. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Address	*<Ref>*[2]	Mandatory. Non-editable selection field. Master data FTP client address. The address object represents an individual FTP user with access to the FTP server specified in *Connection*.
Connection	*<Ref>*[2]	Mandatory. Non-editable selection field. Master data FTP client connection. The connection object represents an FTP server and its communication settings.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- FTP client adapter started.

Result:

- Current process attachment: not accessed and not changed.

Sets the following properties:

- TARGET_FILENAME := The name of the target file.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acForwardingFTP fault.
Directory without permission	Error. Throws acForwardingFTP fault.
Not existing source file	Error. Throws acForwardingFTP fault.
Source file without permission	Error. Throws acForwardingFTP fault.
Locked source file	Error. Throws acForwardingFTP fault.
Already existing target file	Error. Throws acForwardingFTP fault.
File name with unsupported characters	Error. Throws acForwardingFTP fault.
Invalid FTP client connection: wrong host or port	Error. Throws acForwardingFTP fault.
Invalid FTP client address: wrong user	Error. Throws acForwardingFTP fault.
Invalid FTP client address: wrong relative path	Error. Throws acForwardingFTP fault.

Policy: none.

1.12.9 mdFTP

Caption: FTP send.

Throws Fault=acForwardingFTP.

The *acForwardingFTP* activity is a single file activity that sends the current process attachment to an FTP server. This activity actually represents a file upload. Uses the *sendFile* operation from the *FTP Client Adapter* component calling it with a file ???? renaming script (refer to chapter *1.13.7 (Sending Messages)* of the *BIS6 Base Components* manual or to chapter *8.2 (Sending Messages)* of the *FTP Client Adapter* manual for more details).

Field name	Value	Description
File name	*<FileName>*[1]	Mandatory. Editable text field. Empty by default. The name of the target file. Must be a valid file name. The *File name* supports: • Wildcards '*' and '?' enabling generating unique target file names using a special syntax, e.g. 'File-*.tst' leads to 'File-BNHE337K7I.tst'. • Relative paths where the relative path is related to the *Connection* and the *Address* object. The path part does not support any mask patterns. A relative path from *Connection* is appended by this relative path if existing. Path composition depending on the relative paths is described below in Relative path handling. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Address	*<Ref>*[2]	Mandatory. Editable selection field. Default value '<select>'. Master data FTP client address. The address object represents an individual FTP user with access to the FTP server specified in *Connection*. The *Address* supports:

Field name	Value	Description
		- System variables (page 392), e.g. '${MY_FTP_ADDRESS}'. - Properties (page 392), e.g. '$[MY_FTP_ADDRESS]'. XPath expressions are not supported.
Connection	*<Ref>*[2]	Mandatory. Editable selection field. Master data FTP client connection. The connection object represents an FTP server and its communication settings. The *Connection* supports: - System variables (page 392), e.g. '${MY_FTP_CONNECTION}'. - Properties (page 392), e.g. '$[MY_FTP_CONNECTION]'. XPath expressions are not supported.
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Relative path	*<FilePath>*[3]	Optional. Editable text field. Empty by default. Relative path in relation to the *Connection* root path. A relative path from *Connection* is appended by this relative path if existing. Must end with a back slash '\'. The *Relative path* supports: - System variables (page 392), e.g. '${MY_FILE_PATH}'. - Properties (page 392), e.g. '$[MY_FILE_PATH]'. - XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported. Path composition depending on the relative paths is described below in Relative path handling.
Append mode		Optional. Non-editable selection field. Provides three different modes that determine how to handle the case when a file has already been created in the target location. In each case, the target file with the name *File name* is created if it does not yet exist.
	Append	Default. If a file with the name *File name* already exists in the target directory, the new file data will be appended to the existing file.
	Create new file	If a file with the name *File name* already exists in the target directory, the process stops with error.
	Overwrite	If a file with the name *File name* already exists in the target directory, it will be overwritten.
Script		Optional. Non-editable selection field. Provides three different FTP script modes. Affects visibility of *Script file URI* and *Script command* fields.
	No script	Default. Forwards the file to the FTP adapter without any script.
	File	Use the script in the file specified in *Script file URI*.
	Text	Use the script specified in the *Script command* field.
Script file URI	*<URI>*[4]	Mandatory. Editable text field. Only visible when the *File* mode is selected in *Script*. URI to the script file in case of *Script* set to File mode. The *Script file URI* parameter will translate a simple and double backslashes with one forward slash in the gui, the value in the db and bpel process is untouched. The *Script file URI* supports:

Field name	Value	Description
		• System variables (page 392), e.g. '${QA_HOME_DIR}\FTP_Scripts\ftpscript.dta'. • Properties (page 392), e.g. '$[SCRIPT_FILE_URI]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported. Refer to chapter *10 Scripting (link)*in the *FTP Client Adapter* manual for further details about scripts.
Script command	*<Text>*[5]	Mandatory. Editable text field. Script command in case of *Script* set to Text mode. Only visible when the *Text* mode is selected in *Script*. The *Script command* supports: • System variables (page 392), e.g. '${MFT-FTP_SCRIPT}'. • Properties (page 392), e.g. '$[MFT-FTP_SCRIPT]'. Refer to chapter *10 Scripting (link)* in the *FTP Client Adapter* manual for further details about scripts.
Abort on error	True/False	Optional. Refer to Common Fields (page 3).
Log message	*<Text>*[5]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Valid relative file path.
4) Valid URI.
5) Arbitrary text.

Preconditions:

- FTP client adapter started.
- Available attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- TARGET_NAME := The name of the target file.
- TARGET_PATH := The (relative) path of the target file.

Relative path handling

The final path consists of several parts:

- Base directory <BaseDir> in FTP client *Connection*
- Relative path <RelPathAddr> in FTP client *Address*,
- Relative path <RelPathAct1> in this activity in the *File name* field
- Relative path <RelPathAct2> in this activity in the *Relative path* field

Depending on which relative paths in this activity are available, the resulting file path is as follows

- Relative path in *File name* field only, append relative path in *Address*: <BaseDir>/<RelPathAddr>/<RelPathAct1>.
- Relative path in *Relative path* field only, overwrite relative path in *Address*: <BaseDir>/<RelPathAct2>.
- Relative path in both *File name field* and *Relative path* field, overwrite relative path in *Address*: <BaseDir>/<RelPathAct2>/<RelPathAct1>.

Possible cases of failure (ACO denotes the relevant *Append mode*s - A=Append, C=Create new file, O=Overwrite):

Possible case of failure	Append mode	Success or error behavior
Not existing target directory	ACO	Error. Throws acForwardingFTP fault.
Target directory without permission	ACO	Error. Throws acForwardingFTP fault.
Already existing target file	AO	Success. Existing file is being appended (A) or overwritten (O).
	C	Error. Throws acForwardingFTP fault.
Target file without permission	AO	Error. Throws acForwardingFTP fault.
	C	Error. Throws acForwardingFTP fault. The case actually matching is 'Already existing file'.
Locked target file	AO	Error. Throws acForwardingFTP fault.
	C	Error. Throws acForwardingFTP fault. The case actually matching is 'Already existing file'.
Target File name with unsupported characters	ACO	Error. Throws acForwardingFTP fault.
No backslash '\' in *Relative path*	ACO	Error. Throws acForwardingFTP fault.
Not existing script directory	ACO	Error. Throws acForwardingFTP fault.
Script directory without permission	ACO	Error. Throws acForwardingFTP fault.
Not existing script file	ACO	Error. Throws acForwardingFTP fault.
Script file without permission	ACO	Error. Throws acForwardingFTP fault.
Unsupported characters in *Script file URI*	ACO	Error. Throws acForwardingFTP fault.
Not existing *Connection*	ACO	Error. Throws acForwardingFTP fault.
Not existing *Address*	ACO	Error. Throws acForwardingFTP fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/FTP ALLOWANCE/SYSTEM-EXECUTE-SCRIPT

1.13 SFTP Activities

This section describes activities that provide remote file handling on an SFTP server.

1.13.1 acSFTPList

Caption: SFTPList.

Throws Fault=acSFTPList.

The *acSFTPList* activity lists multiple files of a directory on an SFTP server and returns a FILELIST object (CSV file) that can be used as direct input to the *scForEach* (page 10) activity providing a loop element. Uses the *polling* operation from the *SFTP Client Adapter* component setting the initiation mode to *listFiles* (refer to chapter *1.37.4 polling (Polling Messages)* of the *BIS6 Base Components* manual or to chapter *8.3 polling (Poll ing Messages)* of the *SFTP Client Adapter* manual for more details).

Mode of operation:
The *SFTP Client Adapter* component lists the directory for files matching the file mask and returns a list of file names and file paths. This list is converted into CSV format by the *BIC Client* adapter.

Field name	Value	Description
File mask	*<Pattern>*[1]	Mandatory. Editable text field. Default value is '*.*'. Case sensitive file list filter acting as include pattern. Processes only files that match this condition. Actually a download mask for the SFTP server. Overwrites the *Filename* field from the SFTP *Address*. The *File mask* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. • Character sets matching exactly one character of the specified set, e.g. [abc]. • Character ranges matching exactly one character of the specified range, e.g. [a-z]. • Alternatives, e.g. '*.(txt|log)' matching all files '*.txt' and '*.log'. • Relative paths where the relative path is related to the *Connection* base directory. A relative path from *Address* is overwritten. The path part does not support any mask patterns. • System variables (page 392), e.g. '${MY_INCLUDE_PATTERN}'. • Properties (page 392), e.g. '$[MY_INCLUDE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Exclude pattern	*<Pattern>*[2]	Optional. Editable text field. Empty by default. Case sensitive exclusion file mask. Processes only files that do not match this condition. Overwrites the *Global exclude mask* field from the SFTP *Address*. The *Exclude pattern* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. • Character sets matching exactly one character of the specified set, e.g. [abc].

Field name	Value	Description
		- Character ranges matching exactly one character of the specified range, e.g. [a-z]. - Alternatives, e.g. '*.(txt|log)' matching all files '*.txt' and '*.log'. - System variables (page 392), e.g. '${MY_EXCLUDE_PATTERN}'. - Properties (page 392), e.g. '$[MY_EXCLUDE_PATTERN]'. - XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Relative paths are not supported.
Mapping parameter		Optional. Editable text field. Empty by default. Mapping parameter for BIC conversion into CSV format. Available for the mapping as an AdditionalParameter with the name 'PARAMETER'. The *Mapping parameter* supports: - System variables (page 392), e.g. '${MY_MAPPING_PARAM}'. - Properties (page 392), e.g. '$[MY_MAPPING_PARAM]'. - XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Address	*<Ref>*^{3}	Mandatory. Editable selection field. Master data SFTP client address. Represents an individual SFTP user with access to the SFTP server specified in *Connection*. The *Address* supports: - System variables (page 392), e.g. '${SFTP_ADDRESS}'. - Properties (page 392), e.g. '$[SFTP_ADDRESS]'.
Connection	*<Ref>*^{3}	Mandatory. Editable selection field. Master data SFTP client connection. Represents an SFTP server and its communication settings. The *Connection* supports: - System variables (page 392), e.g. '${SFTP_CONNECTION}'. - Properties (page 392), e.g. '$[SFTP_CONNECTION]'.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*^{3}	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*^{4}	Optional. Refer to Common Fields (page 3).

1) Case sensitive SFTP file download mask pattern allowing wildcards.
2) Case sensitive file mask pattern allowing wildcards.
3) Reference to master data.
4) Arbitrary text.

Additional configuration:

System variable	Description
MFT-FILELISTCREATOR	Name of mapping for conversion of file list into FILELIST CSV format. ⚠ Changing this mapping affects all list activities with FILELIST format!

Preconditions:

- SFTP client adapter started.
- BIC client adapter started.
- Mapping from ${MFT-FILELISTCREATOR} system variable available in repository.

Result:

- New attachment: CSV file containing the file list. If no files are matching the file is empty.

Sets the following properties:

- LIST_COUNT := Number of entries in the returned file list.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acSFTPList fault.
Directory without permission	Error. Throws acSFTPList fault.
Empty directory	Success. Returns a list with 0 files.
File without permission	Success. Returns a list with all files.
Locked file	Success. Returns a list with all files.
Invalid SFTP client connection: wrong host or port	Error. Throws acSFTPList fault.
Invalid SFTP client address: wrong user or user authentification (password, key)	Error. Throws acSFTPList fault.
Invalid SFTP client address: wrong relative path	Error. Throws acSFTPList fault.

Policy: none.

1.13.2 acSFTPPickFile

Caption: SFTPPickFile.

Throws Fault=acSFTPPickFile.

The acSFTPPickFile activity is a single file activity that creates a process attachment from the specified file located on an SFTP server. Uses the *polling* operation from the *SFTP Client Adapter* component setting the initiation mode to *pollingResponse* (refer to chapter *1.37.4 polling (Polling Messages)* of the *BIS6 Base Components* manual or to chapter *8.3 polling (Poll ing Messages)* of the *SFTP Client Adapter* manual for more details).

Field name	Value	Description
File name	*<FileName>*[1]	Mandatory. Editable text field. Empty by default. The valid name of the source file. The *File name* supports: • Relative paths where the relative path is related to the *Connection* base directory. A relative path from *Address* is overwritten. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Address	*<Ref>*[2]	Mandatory. Editable selection field. Master data SFTP client address. Represents an individual SFTP user with access to the SFTP server specified in *Connection*. The *Address* supports: • System variables (page 392), e.g. '${SFTP_ADDRESS}'. • Properties (page 392), e.g. '$[SFTP_ADDRESS]'.
Connection	*<Ref>*[2]	Mandatory. Editable selection field. Empty by default. Master data SFTP client connection. Represents an SFTP server and its communication settings. The *Connection* supports: • System variables (page 392), e.g. '${SFTP_CONNECTION}'. • Properties (page 392), e.g. '$[SFTP_CONNECTION]'.
Abort on error	True/False	Refer to Common Fields (page 3).
Delete picked file	True	Delete the picked file.
	False	Default. Keep the picked file.
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- SFTP client adapter started.

Result:

- New attachment: Picked file in case of successful download. Previous attachment in case of no download (see table below).

Sets the following properties:

- FILE_NAME := The name of the source file to pick.
- TARGET_NAME := The name of the source file to pick.
- TARGET_PATH := The path of the source file to pick.

- FILE_COUNT := The number of picked files, 1 in case of real success, 0 in case of error when the *Abort on error* flag is not switched on or in cases of success (see table below).
- PICK_FILE_COUNT := The number of pickable files matching the *File name*.
- FILES_LEFT := The number of files that have not been picked.
- DIRECTION := 'IN'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acSFTPPickFile fault.
Directory without permission	Error. Throws acSFTPPickFile fault.
Empty directory.	Error. Throws acSFTPPickFile fault.
File not found.	Error. Throws acSFTPPickFile fault.
File without permission	Error. Throws acSFTPPickFile fault.
Locked file	Error. Throws acSFTPPickFile fault.
Empty file	Success. File is picked up.
File name with unsupported characters	Error. Throws acSFTPPickFile fault.
Invalid SFTP client connection: wrong host or port	Error. Throws acSFTPPickFile fault.
Invalid SFTP client address: wrong user or password	Error. Throws acSFTPPickFile fault.
Invalid SFTP client address: wrong relative path	Error. Throws acSFTPPickFile fault.

*****) In successful cases, where the target file could not be downloaded, the previous attachment is not replaced by the picked file and is kept untouched as current attachment.

If the file should be deleted after download but the delete operation did not succeed, the fetched file is returned as an attachment into the process.

1.13.3 acSFTPPolling

Caption: SFTPPolling.

Throws Fault=acSFTPPolling.

The *acSFTPPolling* activity is a multiple file activity that polls multiple files from a specified location on an SFTP server that trigger new MFT-SFTPEvents (refer to chapter MFT-SFTPEvent of the *MFT Standard Solution - Adapters* manual for more details). This actually represents a file download. Uses the *polling* operation from the *SFTP Client Adapter* component setting the initiation mode to *separateRequest* (refer to chapter *1.37.4* polling *(Polling Messages)* of the *BIS6 Base Components* manual or to chapter *8.3 polling (Polling Messages)* of the *SFTP Client Adapter* manual for more details).

There are 2 different polling configurations available, one with and one without FTP master data:

Field name	Value	Description
Poll without MD	True	Enter SFTP connection data directly.
	False	Use SFTP client master data.

Poll with SFTP master data:

Field name	Value	Description
Send without MD	False	Use the SFTP client master data.
File mask	*<Pattern>*[1]	Mandatory. Editable text field. Default value is '*.*' matching all available files. Case sensitive file list filter acting as include pattern. Processes only files that match this condition. The *File mask* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. • Character sets matching exactly one character of the specified set, e.g. [abc]. • Character ranges matching exactly one character of the specified range, e.g. [a-z]. • Alternatives, e.g. '*.(txt|log)' matching all files '*.txt' and '*.log'. • Relative paths where the relative path is related to the *Conne ction* base directory. The path part does not support any mask patterns. A relative path from *Address* is overwritten. • System variables (page 392), e.g. '${MY_FILE_PATTERN}'. • Properties (page 392), e.g. '$[MY_FILE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Exclude pattern	*<Pattern>*[1]	Optional. Editable text field. Empty by default. Case sensitive exclusion file mask. Processes only files that do not match this condition. The *Exclude pattern* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. • Character sets matching exactly one character of the specified set, e.g. [abc]. • Character ranges matching exactly one character of the specified range, e.g. [a-z]. • Alternatives, e.g. '*.(txt|log)' matching all files '*.txt' and '*.log'. • System variables (page 392), e.g. '${MY_EXCLUDE_PATTERN}'. • Properties (page 392), e.g. '$[MY_EXCLUDE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Relative paths are not supported.
Address	*<Ref>*[2]	Mandatory. Editable selection field. Master data SFTP client address. Represents an individual SFTP user with access to the SFTP server specified in *Connection*. The *Address* supports: • System variables (page 392), e.g. '${SFTP_ADDRESS}'. • Properties (page 392), e.g. '$[SFTP_ADDRESS]'.
Connection	*<Ref>*[2]	Mandatory. Editable selection field. Master data SFTP client connection. Represents an SFTP server and its communication settings.

Field name	Value	Description
		The *Connection* supports: • System variables (page 392), e.g. '${SFTP_CONNECTION}'. • Properties (page 392), e.g. '$[SFTP_CONNECTION]'.
Delete polled files	True	Delete the polled files.
	False	Keep the polled files.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Case sensitive file mask pattern allowing wildcards.
2) Reference to master data.
3) Arbitrary text.

Poll without SFTP master data:

Field name	Value	Description
Send without MD	True	Directly enter all data. Do not use the SFTP client master data.
File mask	*<Pattern>*[1]	Mandatory. Editable text field. Default value is '*.*' matching all available files. Case sensitive file list filter acting as include pattern. Processes only files that match this condition. The *File mask* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. • Character sets matching exactly one character of the specified set, e.g. [abc]. • Character ranges matching exactly one character of the specified range, e.g. [a-z]. • Alternatives, e.g. '*.(txt|log)' matching all files '*.txt' and '*.log'. • Relative paths where the relative path is related to the *Conne ction* base directory. The path part does not support any mask patterns. A relative path from *Address* is overwritten. • System variables (page 392), e.g. '${MY_FILE_PATTERN}'. • Properties (page 392), e.g. '$[MY_FILE_PATTERN]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Exclude pattern	*<Pattern>*[1]	Optional. Editable text field. Empty by default. Case sensitive exclusion file mask. Processes only files that do not match this condition. The *Exclude pattern* supports: • Wildcard characters '*', representing zero or more characters, and '?', representing exactly one character. • Character sets matching exactly one character of the specified set, e.g. [abc]. • Character ranges matching exactly one character of the specified range, e.g. [a-z]. • Alternatives, e.g. '*.(txt|log)' matching all files '*.txt' and '*.log'.

Field name	Value	Description
		- System variables (page 392), e.g. '${MY_EXCLUDE_PATTERN}'. - Properties (page 392), e.g. '$[MY_EXCLUDE_PATTERN]'. - XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Relative paths are not supported.
Host	<Host>	Mandatory. Editable text field. Empty by default. The host name or IP address of the SFTP server.
Port	*<Int>*[2]	Mandatory. Editable text field. Empty by default. The port of the SFTP server.
Transfer protocol		Mandatory. Non-editable selection field. Provides the expected transfer protocol.
	SCP	Antecedent secure copy protocol. Provides simple file copies only.
	SFTP	Default value. SSH file transfer protocol. Provides secure file transfer and secure file access on a remote file system.
	SSH	Secure shell protocol.
Truststore for known servers	*<Ref>*[3]	Mandatory. Non-editable selection field opening a selection dialog. Empty by default. The trusted keystore that contains the server certificates of the known partner hosts.
Root path		Optional. Editable text field. Empty by default. The user's base directory on the server, relative to the user's home folder.
Delete polled files	True	Delete the polled files.
	False	Keep the polled files.
File sort		Mandatory. Non-editable selection field. Provides available sort modes for resulting file list.
	Alphabetic	All files are sorted alphabetically by file name.
	Last modified	All files are sorted by the time of last modification.
	No sorting	Default value. The files are processed in the order in which they are listed.
Maximum file size		Mandatory. Editable text field. Default value is 0. Maximally allowed file size in bytes. The value 0 means unlimited size. Maximum value is 999999999999999999.
Keep alive	True	Default. Activates the *Keep-alive* mechanism. Refer to chapter *9.3 Keep-alive feature* in the *SFTP Client Adapter* manual for further details.
	False	Does not activate the *Keep-alive* mechanism.
Socket timeout	*<Int>*[2]	Mandatory. Editable text field. Default value is 180. Read message socket timeout in seconds. The value 0 means no timeout.
User	<UserName>	Mandatory. Editable text field. Empty by default. The user name for establishing a connection to the SFTP server.
User authentication mode		Mandatory. Non-editable selection field. Specifies supported user authentication methods (refer to chapter *3.3 User Authentication* in the *SFTP Client Adapter* manual for further details).
	Password	Default value. User password is required. Makes *Password* visible.
	Private key	Private key required. Makes *Key for authentication* visible.

Field name	Value	Description
	Private key and password	Private key and user password required.
Password	<Password>	Optional. Editable password field. Empty by default. The *User* password for establishing a connection to the SFTP server. Visible when *User authentication mode* is switched to 'Password'.
Key for authentication	*<Ref>*[3]	Optional. Non-editable selection field providing a selection dialog. Empty by default. Private key location in KSM. Visible when *User authentication mode* is switched to 'Private key'.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[3]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Case sensitive file mask pattern allowing wildcards.
2) Positive integer value denoting a valid port number.
3) Reference to master data.
4) Arbitrary text.

Preconditions:

- SFTP client adapter started.

Result:

- New *MFT-SFTPEvents* for the polled files.
- Current attachment not accessed and not replaced.

Sets the following properties:

- FILE_COUNT := The number of polled files.
- FILES_LEFT := The number of files that could not be polled.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acSFTPPolling fault.
Directory without permission	Error. Throws acSFTPPolling fault.
Empty directory	Success. FILE_COUNT is set to 0.
Empty file	Success. FILE_COUNT is increased.
Files with and without permission	Success. For each file without permission: file is not polled, FILES_LEFT is increased.
Only files without permission	Success. No file is polled, FILE_COUNT is set to 0. FILES_LEFT is increased.
Locked files and files with permission	Success. For each locked file: file is not polled, FILES_LEFT is increased.
Only locked files	Success. No file is polled, FILE_COUNT is set to 0. FILES_LEFT is increased.
Invalid SFTP client connection: wrong host or port	Error. Throws acSFTPPickFile fault.
Invalid SFTP client address: wrong user or password	Error. Throws acSFTPPickFile fault.
Invalid SFTP client address: wrong relative path	Error. Throws acSFTPPickFile fault.

Policy: none.

1.13.4 acSFTPRemoveFile

Caption: SFTPRemoveFile.

Throws Fault=acSFTPRemoveFile.

The acSFTPRemoveFile activity is a single file activity that removes a file located on an SFTP server. Uses the *removeFile* operation from the *SFTP Client Adapter* component setting the initiation mode to *separateRequest* (refer to chapter *1.37.6 removeFile* of the *BIS6 Base Components* manual or to chapter *8.5 removeFile* of the *SFTP Client Adapter* manual for more details).

Field name	Value	Desription
File mask	<FileName>[1]	Mandatory. The valid name of the source file. The *File name* supports: • Relative paths where the relative path is related to the *Connection* base directory. A relative path from *Address* is overwritten. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Address	*<Ref>*[2]	Mandatory. Not editable selection field. Master data SFTP client address. Represents an individual SFTP user with access to the SFTP server specified in *Connection*.
Connection	*<Ref>*[2]	Mandatory. Not editable selection field. Master data SFTP client connection. Represents an SFTP server and its communication settings.
Abort on error	True/False	Refer to Common Fields (page 3).

Field name	Value	Desription
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- SFTP client adapter started.

Result:

- Current attachment not accessed and not replaced.

Sets the following properties:

- TARGET_NAME := The name of the removed file.
- TARGET_PATH := The path of the removed file.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acSFTPRemoveFile fault.
Directory without permission	Error. Throws acSFTPRemoveFile fault.
Not existing file	Error. Throws acSFTPRemoveFile fault.
Empty file	Success. File is removed.
File without permission	Error. Throws acSFTPRemoveFile fault. I Different behavior when using the BIS SFTP controller: Success. File is removed.
Locked file	Error. Throws acSFTPRemoveFile fault.
File name with unsupported characters	Error. Throws acSFTPRemoveFile fault.

Policy: none.

1.13.5 acSFTPRename

Caption: SFTPRename.

Throws Fault=acSFTPRename.

The *acSFTPRename* activity is a single file activity that allows an existing remote source file to be renamed to a remote target file on an SFTP server. Uses the *renameFile* operation from the *SFTP Client Adapter* component setting the initiation mode to *separateRequest* (refer to chapter *1.37.5 renameFile* of the *BIS6 Base Components* manual or to chapter *8.6 renameFile* of the *SFTP Client Adapter* manual for more details).

Field name	Value	Description
Remote file name	*<FileName>*[1]	Mandatory. Editable text field. Default value 'MyFile.txt'. The name of the target file. Must be a valid file name. Supports: • Relative paths where the relative path is related to the *Connection* and the *Address* object. The path part does not support any mask patterns. A relative path from *Connection* is appended by this relative path if existing. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Target file name	*<FileName>*[1]	Mandatory. Editable text field. Empty by default. The name of the target file. Must be a valid file name. Supports: • Relative paths where the relative path is related to the *Connection* and the *Address* object. The path part does not support any mask patterns. A relative path from *Connection* is appended by this relative path if existing. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Address	*<Ref>*[2]	Mandatory. Editable selection field. Master data SFTP client address. Represents an individual SFTP user with access to the SFTP server specified in *Connection*. Supports: • System variables (page 392), e.g. '${CLIENT_ADDRESS}'. • Properties (page 392), e.g. '$[CLIENT_ADDRESS]'.
Connection	*<Ref>*[2]	Mandatory. Editable selection field. Master data SFTP client connection. Represents an SFTP server and its communication settings. Supports: • System variables (page 392), e.g. '${CLIENT_CONNECTION}'. • Properties (page 392), e.g. '$[CLIENT_CONNECTION]'.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- SFTP client adapter started.

Result:

- Current attachment not accessed and not replaced.

Sets the following properties:

- TARGET_FILENAME := The name of the target file.

Possible cases of failure:

Possible case of failure	Success or error behavior
Not existing directory	Error. Throws acForwardingSFTP fault.
Directory without permission	Error. Throws acForwardingSFTP fault.
Not existing source file	Error. Throws acForwardingSFTP fault.
Source file without permission	Error. Throws acForwardingSFTP fault.
Locked source file	Error. Throws acForwardingSFTP fault.
Already existing target file	Error. Throws acForwardingSFTP fault.
File name with unsupported characters	Error. Throws acForwardingSFTP fault.
Invalid SFTP client connection: wrong host or port	Error. Throws acForwardingSFTP fault.
Invalid SFTP client address: wrong user	Error. Throws acForwardingSFTP fault.
Invalid SFTP client address: wrong relative path	Error. Throws acForwardingSFTP fault.
Non-existing system variable	Error. Throws acForwardingSFTP fault.
Non-existing property in *Address* or *Connection*	Error. Throws acForwardingSFTP fault.
Non-existing property in Target file name	Success. Used as target file name.

Policy: None.

1.13.6 mdSFTP

Caption: SFTP send.

Throws Fault=acForwardingSFTP.

The *acForwardingSFTP* activity is a single file activity that allows the current attachment to be sent to an SFTP server. This activity actually represents a file upload. Uses the *sendFile* operation from the *SFTP Client Adapter* component setting the initiation mode to *separateRequest* (refer to chapter *1.37.7 sendFile (Sending Messages)* of the *BIS6 Base Components* manual or to chapter *8.2 sendFile (Sending Messages)* of the *SFTP Client Adapter* manual for more details).

Field name	Value	Description
Address	*<Ref>*[2]	Mandatory. Master data SFTP client address. Represents an individual SFTP user with access to the SFTP server specified in *Connection*. Editable selection field. Supports:

Field name	Value	Description
		• System variables (page 392), e.g. '${MY_SFTP_ADDRESS}'. • Properties (page 392), e.g. '$[MY_SFTP_ADDRESS]'. XPath expressions are not supported.
Connection	*<Ref>*[2]	Mandatory. Master data SFTP client connection. Represents an SFTP server and its communication settings. Editable selection field. Supports: • System variables (page 392), e.g. '${MY_SFTP_CONNECTION}'. • Properties (page 392), e.g. '$[MY_SFTP_CONNECTION]'. XPath expressions are not supported.
File name	*<FileName>*[1]	Mandatory. The target file name to be used on the SFTP server. The *File name* supports: • Relative paths where the relative path is related to the *Connecti on* base directory. A relative path from *Address* is overwritten. If *Relative path* is filled, the complete *File name* is appended. The path part does not support any mask patterns. • Blanks in the file name. • German umlauts if the *Connection* Filename encoding allows this.. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Upload option		Provides three modes of filename spelling for the target file. Overwrites Filename spelling field in *Connection*.
	keep original	Default. Keep the target *File name* unchanged.
	lower-case	Change the target *File name* to lower case.
	upper-case	Change the target *File name* to upper case.
Append mode		Provides four different modes that determine how to handle the case when a file has already been created in the target location. In each case, the target file with the name *File name* is created if it does not yet exist. Overwrites File Transfer mode field in *Connection*.
	Append	If a file with the name *File name* already exists in the target directory, the new file data will be appended to the existing file.
	Fail if exists	When the target file already exists, the activity stops with error and throws an *acForwardingSFTP* fault.
	overwrite file	Default. If a file with the name *File name* already exists in the target directory, it will be overwritten.
	unique filename	When the target file already exists, a new file is created whose name is combined from the target *File name* and a UUID, for instance `test _d1c28c62-c5c7-11e5-a87e-6f760a0069c5 .dta .`
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).

Field name	Value	Description
Relative path	*<FilePath>*[3]	Optional. Relative path in relation to the *Connection* base directory. A relative path from *Address* is overwritten by this relative path if existing. Editable field. Can end with a slash '/' or a back slash '\' but this is not necessary. The *Relative path* supports: • System variables (page 392), e.g. '${MY_FILE_PATH}'. • Properties (page 392), e.g. '$[MY_FILE_PATH]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Reference to master data.
3) Valid relative file path.
4) Arbitrary text.

Preconditions:

- SFTP client adapter started.
- Available process attachment.

Result:

- Process attachment not replaced.

Sets the following properties:

- TARGET_NAME := The name of the target file.
- TARGET_PATH := The (relative) path of the target file.

Possible cases of failure (ACO denotes the relevant *Append modes* - A=Append, F=Fail if exists, O=Overwrite file, U=Unique filename):

Possible case of failure	Append mode	Success or error behavior
Not existing target directory	AFOU	Error. Throws acForwardingSFTP fault.
Target directory without permission	AFOU	Error. Throws acForwardingSFTP fault.
Already existing target file	AOU	Success. Existing file is being appended (A) or overwritten (O) or stored with a unique name (U).
	F	Error. Throws acForwardingSFTP fault.
Target file without permission	OU	Success. Existing file is overwritten (O) or stored with a unique name (U).

Possible case of failure	Append mode	Success or error behavior
	A	Error. Throws acForwardingSFTP fault.
	F	Error. Throws acForwardingSFTP fault. The case actually matching is 'Already existing file'.
Locked target file	U	Success. Existing file is stored with a unique name (U).
	AO	Error. Throws acForwardingSFTP fault.
	F	Error. Throws acForwardingSFTP fault. The case actually matching is 'Already existing file'.
Target *File name* with unsupported characters	AFOU	Error. Throws acForwardingSFTP fault.
Relative path without slash or back slash	AFOU	Success (except for other table failure items matching).
Not existing SFTP client connection (no master data match)	AFOU	Error. Throws acForwardingSFTP fault.
Invalid SFTP client connection: wrong host or port	AFOU	Error. Throws acForwardingSFTP fault.
Not existing SFTP client address (no master data match)	AFOU	Error. Throws acForwardingSFTP fault.
Invalid SFTP client address: wrong user or user authentification (password, key)	AFOU	Error. Throws acForwardingSFTP fault.
Invalid SFTP client address: wrong relative path	AFOU	Error. Throws acForwardingFTP fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/SFTP

1.13.7 acSSHRemoteShell

Caption: SSHRemoteShell.

Throws Fault=acSSHRemoteShell.

The acSSHRemoteShell activity is an activity that executes a single command via ssh on an SFTP server. Uses the *executeCommand* operation from the *SFTP Client Adapter* component setting the initiation mode to *separateRequest* (refer to chapter *1.37.1 executeCommand* of the *BIS6 Base Components* manual or to chapter *8.7 executeCommand* of the *SFTP Client Adapter* manual for more details).

Field name	Value	Description
Address	*<Ref>*[1]	Mandatory. Not editable selection field. Master data SFTP client address. Represents an individual SFTP user with access to the SFTP server specified in *Connection*.
Connection	*<Ref>*[1]	Mandatory. Not editable selection field. Master data SFTP client connection. Represents an SFTP server and its communication settings.

Field name	Value	Description
Command	<Script>[2]	Mandatory. The shell command to be executed on the SFTP server. The *Command* supports: • System variables (page 392), e.g. '${MY_REMOTE_SHELL_SCRIPT}'. Supports a shell script. • Properties (page 392), e.g. '$[MY_FILE_PATTERN]'. I A shell script can be executed on the SFTP server via 2 ways: Either use a system variable. Or use the command “cat | sh” and provide the script in the current process attachment. Note that *Send attachment* has to be switched on in this case.
Send attachment	True	Provide current process attachment to the remote shell. (Written to stdin.)
	False	Do not use current process attachment.
Console output as result	True	Process attachment is set to the attachment returned by the SFTP response. Generally, this is the console output created by the *Comm and*. If there is no such attachment, an empty attachment is created. (i.e. ls -alt, find . * | grep 'D')
	False	No attachment as activity result. The current process attachment is not changed.
ENV1		Mandatory. Default value is 'NOT_SET'. Environment variable value. Set as environment variable in the remote shell.
ENV2		Mandatory. Default value is 'NOT_SET'. Environment variable value. Set as environment variable in the remote shell.
ENV3		Mandatory. Default value is 'NOT_SET'. Environment variable value. Set as environment variable in the remote shell.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Valid remote shell script.
3) Arbitrary text.

Preconditions:

- SFTP client adapter started.
- Available process attachment if *Send attachment* is switched on.

Result:

- New process attachment if *Console output as result* is switched on:
  if an attachment is available as command response this is used.
  If no attachment is available, an empty attachment is created.
- The current process attachment is not changed if *Console output as result* is switched off.

Sets the following properties: none

A typical use case for acRemoteShell is unzipping a file on the server side after transport (e.g. via SFTP).

1.14 HTTP Activities

This section describes HTTP related activities.

1.14.1 acHTTPSyncReply

Caption: Set HTTP reply message

Throws Fault=acHTTPSyncReply.

The *acHTTPSyncReply* activity is an activity that creates the HTTP response messages for synchronous HTTP file transfer requests.

Mode of operation:
The *acHTTPSyncReply* activity has to be used as a part of the MFT-HTTPSyncServiceEvent process (refer to chapter *MFT-HTTPSyncServiceEvent (link)* in the *MFT Standard Solution - Adapters* manual for more details), i.e. the Criteria *From adapter* field has to set to 'HTTP synchon (Server)' (internal value is 'SHCSYN'). It creates the HTTP response for the received synchronous HTTP request which is sent to the HTTP request sender. After this *acHTTPSyncReply* activity, the process finishes and ignores any probably succeeding activities. Thus, it is recommended to always use this *acHTTPSyncReply* activity as the last activity in an entity.

Field name	Value	Description
Response type		Mandatory. Non editable selection field. Provides attachment configuration modes.
	Attachment	Default value. Add the current attachment to the HTTP response.
	Attachments from bookmark	Add the bookmarked attachments specified in *Bookmarks* to the HTTP response. Makes *Bookmarks for http reply* visible.
	Plain	Do not add an attachment to the HTTP response.
Content type		Optional. Editable selection field. MIME content types for attachment/s. Supports: • Properties (page 392), e.g. '$[CONTENT_TYPE]'. • System variables (page 392), e.g. '${CONTENT_TYPE}'.
	application/pdf	MIME content-type. PDF file.
	application/zip	MIME content-type. Zip file.
	image/bmp	MIME content-type. BMP file.
	image/tiff	MIME content-type. TIFF file.
	text/csv	MIME content-type. CSV file.
	text/html	MIME content-type. HTML file.
Bookmarks	*<BookmarkList>*[1]	Mandatory. Editable text field. Empty by default. List of bookmark names. Each bookmark is added as an attachment to the HTTP response. Supports: • Properties (page 392), e.g. '$[BOOKMARK_LIST]'.

Field name	Value	Description
		• System variables (page 392), e.g. '${BOOKMARK_LIST}'. • Maximum number of bookmarks is 10. Additional bookmarks are ignored. Only visible if *Response type* is set to 'Attachments from bookmark'.
Status code		Mandatory. Editable selection field. Provides a bunch of predefined status codes, actually a part of HTTP response status codes. The selected status code is added to the HTTP response as HTTP response status. Supports: • Properties (page 392), e.g. '$[HTTP_STATUS_CODE]'. • System variables (page 392), e.g. '${HTTP_STATUS_CODE}'.
	200 OK	Default. Standard response for successful HTTP requests.
	201 Created	Request fulfilled, a new resource has been created.
	202 Accepted	Request accepted for processing, processing not yet completed.
	203 Non-Authoritive Information	Server is a transforming proxy, received a 200 OK, but returns a modified version.
	204 No Content	Request successfully processed, no content returned.
	205 Reset Content	Request successfully processed, no content returned. Unlike 204, this response requires the requester to reset the document view.
	206 Partial Content	Only part of the resource is delivered due to a range header.
	400 Bad Request	Request cannot be processed due to an apparent client error.
	401 Unauthorized	Authorization is required and has failed or has not yet been provided.
	402 Payment Required	Usually unused.
	403 Forbidden	Valid request, but server refuses to respond due to e.g. missing permission.
	404 Not Found	Requested resource not found.
	405 Method Not Allowed	Request method not supported for requested resource.
	406 Not Acceptable	Requested resource can only generated content not accectable according to request accept headers.
Key=Value1	*<KeyValue>*[2]	Optional. Editable text field. Empty by default. Custom header property. First key-value pair that can be added to the HTTP response message header. Supports: • Properties (page 392), e.g. '$[MY_VALUE]' or 'Doctype= $[DOCTYPE]'. • System variables (page 392), e.g. '${MY_VALUE}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).

Field name	Value	Description
Key=Value2	*<KeyValue>*[2]	Optional. Editable text field. Second key-value pair that can be added to the HTTP response message header. Refer to *Key=Value1* above for more details.
Key=Value3	*<KeyValue>*[2]	Optional. Editable text field. Third key-value pair that can be added to the HTTP response message header. Refer to *Key=Value1* above for more details.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Semicolon separated list of bookmark names.
2) Key-value pair with notation: MyKey=MyValue.
3) Arbitrary text.

Preconditions:

- The Criteria must match the *MFT-HTTPSyncServiceEvent (link)* event, i.e. the *From adapter* field has to set to 'HTTP synchon (Server)'.
- HTTP Client adapter started.
- HTTP Controller adapter started.
- Available process attachment: in case of *Response type* set to 'Attachment'.
- Specified bookmarks: in case of *Response type* set to 'Attachments from bookmark'.

Result:

- Current process attachment: not changed.

Sets the following properties:

- HTTP_INTERFACE := 'syncReply'.
- HTTP_REPLY := created HTTP response message containing Status code and Key=Value* if filled and one or more attachments in case of the Attachment and Attachments from bookmark *Response types*.
- RESULT := 'FINISHED_SUCCESS'.
- STATUS := Number from *Status code*.

Possible cases of failure:

Possible case of failure	Response type	Success or error behavior
Criteria *From adapter* field not set to 'HTTP synchon (Server)'		Success. No reply sent to HTTP request sender.
Invalid bookmark	Bookmarks	Error. Throws acHTTPSyncReply fault.
Invalid *Key=Value** syntax		Success. No key and value added to HTTP response.
Invalid *Status code* without number		Success. The status text is used as HTTP response status. The **receiver fails** with a NumberFormatException since a number has been expected. Each retry on the receiver process (actually the HTTP request sender) will send a new HTTP request and initiate a new *MFT -HTTPSyncServiceEvent (link)* process like this one continuing to send the wrong status code.

Policy: none.

Example: How To HTTP Sync Reply (page 418).

1.14.2 mdHTTP

Caption: HTTP send.

Throws Fault=acForwardingHTTP.

The *mdHTTP* activity is a single file activity that sends the current process attachment to an HTTP server. Uses the *sendFile* operation from the *HTTP Client Adapter* component setting the initiation mode to *separateRequest* (refer to chapter *1.18.3 sendFile (Sending Messages)* of the *BIS6 Base Components* manual or to chapter *8.1 sendFile (Sending Messages)* of the *HTTP Client Adapter* manual for more details).

There are 3 different types of configuration available, two with and one without HTTP master data:

Field name	Value	Description
Master data usage type		Mandatory. Non-editable selection field. Supported modes with and without master data.
	Enrich master data with dynamic properties	Use master data enriched with encoding type and content type.
	No master data	Do not use master data.
	Only master data	Default value. Use master data.

Identical fields for all configuration types:

Field name	Value	Description
Logical file name	*<Name>*[1]	Optional. Editable text field. Empty by default. Logical file name for HTTP request. Supports: • System variables (page 392), ${LOGICAL_NAME}'. • Properties (page 392), e.g. '$[LOGICAL_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')).
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Send method		Mandatory. Non-editable selection field. HTTP method used to send a file or an HTTP request to an HTTP server. Refer to *HTTP methods* for more details.
	PUT	Default value. HTTP PUT method.
	POST	HTTP POST method.
	GET	HTTP GET method.
	DELETE	HTTP DELETE method.
...	...	Fields depending on selected *Master data usage type*. See below.
Proxy mode		Optional. Non-editable selection field. Supported proxy modes.
	DT_RROXY_MODE_DEFAULT	Default proxy. The default proxy for the set HTTP protocol.
	DT_RROXY_MODE_GROUP	A proxy group record with an alias. Makes *Proxy group* visible.
	DT_RROXY_MODE_NONE	Default. No proxy.

Field name	Value	Description
	DT_RROXY_MODE_SPECIFIC	A specific proxy server with an alias. Makes *Proxy alias* visible.
Proxy alias	<ProxyName>	Mandatory. Non-editable selection field. Master data proxy group ID. Visible when *Proxy mode* is set to 'DT_PROXY_MODE_GROUP'.
Proxy group	<ProxyName>	Mandatory. Non-editable selection field. Master data proxy server ID. Visible when *Proxy mode* is set to 'DT_PROXY_MODE_SPECIFIC'.
Header parameter	*<KVPair>*[3]	Optional. Editable text field. Additional header parameter. Supports: • System variables (page 392), ${KVPAIR1}'. • Properties (page 392), e.g. '$[KVPAIR1]' or 'PARAM1= $[VALUE1]'. • XPath expressions (page 392), e.g. $(upper-case('Key=value')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')).
Header parameter	*<KVPair>*[3]	Optional. Editable text field. Additional header parameter. Refer to first *Header parameter* above for more details.
Header parameter	*<KVPair>*[3]	Optional. Editable text field. Additional header parameter. Refer to first *Header parameter* above for more details.
Header parameter	*<KVPair>*[3]	Optional. Editable text field. Additional header parameter. Refer to first *Header parameter* above for more details.
Header parameter	*<KVPair>*[3]	Optional. Editable text field. Additional header parameter. Refer to first *Header parameter* above for more details.
Abort on error	True/False	Refer to Common Fields (page 1).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Arbitrary name.
2) Reference to master data.
3) Key-value pair.
4) Arbitrary text.

HTTP Send File with Master Data

Master data usage types 'Only master data' and 'Enrich master data with dynamic properties'.

Field name	Value	Description
Receiver	*<Ref>*[1]	Mandatory. Editable selection field. HTTP client address master data. Holds URI and HTTP header settings. Supports: • System variables (page 392), ${HTTP_RECEIVER}'. • Properties (page 392), e.g. '$[HTTP_RECEIVER]'.
Connection	*<Ref>*[1]	Mandatory. Editable selection field. HTTP client connection master data. The connection data for the HTTP server. Supports: • System variables (page 392), ${HTTP_CONNECTION}'. • Properties (page 392), e.g. '$[HTTP_CONNECTION]'.
Encoding type		Mandatory. Non-editable selection field. Provides supported encoding types. Overwrites the encoding type from the *Receiver* address. Only visible with *Master data usage type* set to 'Enrich master data with dynamic properties'.
	<platform>	Use platform encoding.
	<binary>	Use binary if the encoding is unknown.
	UTF-8	Default. International encoding. ISO/IEC-10646 norm. 8 bit character width minimum.
	<CharSets>[2]	A bunch of additionally supported character sets: US-ASCII, Windows-1252, ISO-8859-1, SeeCpEBCDIC, SeeCP500, ISO-8859-15, UTF-16, UTF-16BE, UTF-16LE, UnicodeBig, UnicodeLittle, windows-1250, windows-1251, windows1253, windows-1257, ISO-8859-2, ISO-8859-4, ISO-8859-5, ISO-8859-7, ISO-8859-9, ISO-8859-13,KOI8-R, windows-1255, windows-1256, windows-1258, windows-31j, EUC-JP, x-EUC-JP-LINUX, Shift_JIS, ISO-2022-JP, x-mswin-936, GB18030, x-EUC-CN, GBK, ISCII91, x-windows-949, EUC-KR, ISO-2022-KR, x-windows-950, x-MS950-HKSCS, x-EUC-TW, Big5, Big5-HKSCS, TIS-620
Content type		Mandatory. Non-editable selection field. Provides the supported MIME content types. Only visible with *Master data usage type* set to 'Enrich master data with dynamic properties'.
	application/json	MIME content type. JSON file.
	application/xml	Default. MIME content type. XML file.
	multipart/form-data	MIME content type. Values submitted through a form.
	text/html	MIME content type. HTML file.
	text/xml	MIME content type. XML file.

1) Reference to master data.
2) Character set/ encoding type from the list.

Preconditions:

- HTTP Client adapter started.
- Available process attachment.

Result:

- New process attachment: only if *Send method* is set to 'GET'.

Sets the following properties:

- TARGET_NAME := Value of *Logical file name*.
- TARGET_PATH := URL path of *Receiver* address.
- DIRECTION := 'OUT'.
- TO_PATH := 'NOT_SET'.
- HTTP_METHOD := Value of *Send method*, one of the following: 'POST', 'PUT', or 'GET'.
- HTTP_REPLY_AVAILABLE := 'TRUE' if the http reply contains an attachment. 'FALSE' otherwise.
- HTTP_REPLY_BOOKMARK := bookmarked response of the HTTP controller in XML format in case HTTP_REPLY_AVAILABLE is 'TRUE'.
- HTTP_REPLY_PROPERTIES := Properties of HTTP response, for instance: 'Date=Mon, 11 Jul 2016 14:24:02 GMT;Server=BIS6.5.2SP24;Content-Type=text/html;'.
- HTTP_STATUS := '200' (OK) in case of success.
- STATUS := 'SUCCESS_NOTIFY' in case of success.

In cases of error:

- FAILED_HTTP_REPLY_BOOKMARK := bookmarked fault attachment in error cases if available, *Abort on error* switched on.
- ACTIVITY_FAULT_BOOKMARK := bookmarked fault attachment in error cases if available, *Abort on error* switched off.

Possible cases of failure:

Possible case of failure	Success or error behavior
HTTP Client adapter not started	Error. Throws acForwardingHTTP fault.
Invalid *Connection*	Error. Throws acForwardingHTTP fault.
Invalid *Receiver* address.	Error. Throws acForwardingHTTP fault.
Invalid *Receiver* URL.	Error. Throws acForwardingHTTP fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/HTTP

HTTP Send File without Master Data

Master data usage type 'No master data'.

Field name	Value	Description
HTTP protocol		Mandatory. Non-editable selection field. Applied HTTP protocol type.
	HTTP	Default value.Standard protocol.

Field name	Value	Description
	HTTPS	Protocol for secure communication. Makes the *Truststore* field visible.
URI	<RelURL>	Mandatory. Editable text field. Empty by default. The URL of the file, the servlet or the service which is addressed with this HTTP server request. Relative path to the desired file or servlet, e.g. '/ SeeburgerHTTP/HTTPController'. Supports: • System variables (page 392), ${URI}'. • Properties (page 392), e.g. '$[URI]'.
Servername	<Host>	Mandatory. Editable text field. Empty by default. The host name or IP address of the HTTP server. Supports: • System variables (page 392), ${SERVER_NAME}'. • Properties (page 392), e.g. '$[SERVER_NAME]'.
Serverport	*<Int>*[1]	Mandatory. Editable text field. Empty by default. The listening port of the HTTP server. Supports: • System variables (page 392), ${SERVER_PORT}'. • Properties (page 392), e.g. '$[SERVER_PORT]'.
Truststore	*<Ref>*[2]	Mandatory. Editable text field text field with key-alias selection dialog. SSL alias from the Keystore Manager. Supports: • System variables (page 392), ${ALIAS_NAME}'. • Properties (page 392), e.g. '$[ALIAS_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('MyFileName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Only visible when *HTTP protocol* is switched to *HTTPS*.
User	<UserName>	Optional. Editable text field. The user name for the Basic Authentication. Supports: • System variables (page 392), ${USER_NAME}'. • Properties (page 392), e.g. '$[USER_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('MyUserName')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Password	<Password>	Optional. Editable password field. The password for the Basic Authentication. Properties, system variables and XPath expressions are not supported.

1) Positive integer value denoting a valid port number.
2) Reference to master data.

Preconditions:

- HTTP client adapter started.

- Available process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- TARGET_NAME := Value of *Logical file name*.
- TARGET_PATH := URL path of *Receiver* address.
- DIRECTION := 'OUT'.
- TO_PATH := 'NOT_SET'.
- HTTP_METHOD := Value of *Send method*, one of the following: 'POST', 'PUT', or 'GET'.
- HTTP_REPLY_AVAILABLE := 'TRUE' if the http reply contains an attachment. 'FALSE' otherwise.
- HTTP_REPLY_BOOKMARK := bookmarked response of the HTTP controller in XML format in case HTTP_REPLY_AVAILABLE is 'TRUE'.
- HTTP_REPLY_PROPERTIES := Properties of HTTP response, for instance: 'Date=Mon, 11 Jul 2016 14:24:02 GMT;Server=BIS6.5.2SP24;Content-Type=text/html;'.
- HTTP_STATUS := '200' (OK) in case of success.
- STATUS := 'SUCCESS_NOTIFY' in case of success.

In cases of error:

- FAILED_HTTP_REPLY_BOOKMARK := bookmarked fault attachment in error cases if available, *Abort on error* switched on.
- ACTIVITY_FAULT_BOOKMARK := bookmarked fault attachment in error cases if available, *Abort on error* switched off.

Possible cases of failure:

Possible case of failure	Success or error behavior
HTTP Client adapter not started	Error. Throws acForwardingHTTP fault.
Invalid *Truststore* alias	Error. Throws acForwardingHTTP fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/HTTP

1.14.3 acSyncReply

Caption: SyncReply.

Throws Fault=acSyncReply.

⚠ **Deprecated! Please use *acHTTPSyncReply* (page 238) and *acGenericServiceReply* (page 365) instead!**

The *acSyncReply* activity is a multiple mode activity that creates response messages for several adapter types.

Field name	Value	Description
Adapter type		Mandatory. Non editable selection field. Provides supported adapter types for synchronous replies.
	Sync http	Creates a HTTP response message. Makes the *Response type* visible. ⚠ **Deprecated! Please use *acHTTPSyncReply* (page 238) instead!**
	Webservice	Creates a Webservice reply. ⚠ **Deprecated beta version! Do not use! Please use *acGenericServiceReply* (page 365) instead!**

Sync http adapter type:

This *acSyncReply* activity type is an activity that creates the HTTP response messages for synchronous HTTP file transfer requests.

Mode of operation:
This *acHTTPSyncReply* activity type has to be used as a part of the MFT-HTTPSyncServiceEvent process, i.e. the Criteria *From adapter* field has to set to 'HTTP synchon (Server)' (internal value is 'SHCSYN'). It creates the HTTP response for the received synchronous HTTP request which is sent to the HTTP request sender. After this *acHTTPSyncReply* activity type, the process finishes and ignores any probably succeeding activities. Thus, it is recommend to always use this *acHTTPSyncReply* activity type as the last activity in an entity.

⚠ **Deprecated! Please use *acHTTPSyncReply* (page 238) instead!**

Field name	Value	Description
Response type		Mandatory. Non editable selection field. Provides attachment configuration modes. Only visible if *Adapter type* set to 'Sync http'.
	Attachment	Default. Add the current attachment to the response message.
	Attachments from bookmark	Add the bookmarked attachments to the response message. Makes *Bookmarks for http reply* visible.
	Plain	Does not add the current attachment to the HTTP response.
Bookmarks for http reply	*<BookmarkList>*[1]	Mandatory. Editable text field. Empty by default. Each bookmark is added as an attachment to the HTTP response. Supports: • Maximum number of bookmarks is 2000. Additional bookmarks are ignored. Only visible if *Response type* is set to 'Attachments from bookmark'.
Status code		Mandatory. Non editable selection field. Provides a bunch of status codes, actually a part of HTTP response status codes. The selected status code is added to the HTTP response as HTTP response status.
	200 OK	Default. Standard response for successful HTTP requests.
	201 Created	Request fulfilled, a new resource has been created.

Field name	Value	Description
	202 Accepted	Request accepted for processing, processing not yet completed.
	203 Non-Authorative Information	Server is a transforming proxy, received a 200 OK, but returns a modified version.
	204 No Content	Request successfully processed, no content returned.
	205 Reset Content	Request successfully processed, no content returned. Unlike 204, this response requires the requester to reset the document view.
	206 Partial Content	Only part of the resource is delivered due to a range header.
	400 Bad Request	Request cannot be processed due to an apparent client error.
	401 Unauthorized	Authorization is required and has failed or has not yet been provided.
	402 Payment Required	Usually unused.
	403 Forbidden	Valid request, but server refuses to respond due to e.g. missing permission.
	404 Not Found	Requested resource not found.
	405 Method Not Allowed	Request method not supported for requested resource.
	406 Not Acceptable	Requested resource can only generated content not accectable according to request accept headers.
keyValue1	*<KeyValue>*^{2}	Optional. Editable text field. Custom header property. First key-value pair that can be added to the HTTP response message header. Supports: • Properties (page 392), e.g. '$[MY_VALUE]' or 'Doctype= $[DOCTYPE]'. • System variables (page 392), e.g. '${MY_VALUE}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
keyValue2	*<KeyValue>*^{2}	Optional. Editable text field. Second key-value pair that can be added to the HTTP response message header. Refer to *Key=Value1* for more details.
keyValue3	*<KeyValue>*^{2}	Optional. Editable text field. Third key-value pair that can be added to the HTTP response message header. Refer to *Key=Value1* for more details.
logmessage	*<Text>*^{3}	Optional. Refer to Common Fields (page 3).

1) Semicolon separated list of bookmark names.
2) Key-value pair with notation: MyKey=MyValue.
3) Arbitrary text.

Preconditions:

- The Criteria must match the MFT-HTTPSyncServiceEvent event, i.e. the *From adapter* field has to set to 'HTTP synchon (Server)'.
- HTTP Client adapter started.

- HTTP Controller adapter started.
- Available process attachment: in case of *Response type* set to 'Attachment'.
- Specified booksmarks: in case of *Response type* set to 'Attachments from bookmark'.

Result:

- Current process attachment: not changed.

Sets the following properties:

- HTTP_INTERFACE := 'syncReply'.
- HTTP_REPLY := created HTTP response message containing Status code and Key=Value* if filled and one or more attachments in case of the Attachment and Attachments from bookmark *Response types*.
- RESULT := 'FINISHED_SUCCESS'.
- STATUS := Number from *Status code*.

Possible cases of failure:

Possible case of failure	**Response type**	**Success or error behavior**
Criteria *From adapter* field not set to 'HTTP synchon (Server)'		Success. No reply sent to HTTP request sender.
Invalid bookmark	Bookmarks	Error. Throws acSyncReply fault.
Invalid *Key=Value** syntax		Success. No key and value added to HTTP response.
Invalid *Status code* without number		Success. The status text is used as HTTP response status. The **receiver fails** with a NumberFormatException since a number has been expected. Each retry on the receiver process (actually the HTTP request sender) will send a new HTTP request and initiate a new MFT-HTTPSyncServiceEvent process like this one continuing to send the wrong status code.

Policy: none.

Example: MFTHTTPSyncReplyHowTo (page 418).

Webservice adapter type:

This *acSyncReply* activity type is an activity that creates the Webservice response messages for synchronous Webservice requests.

⚠ **Deprecated beta version! Do not use! Please use *acGenericServiceReply* (page 365) instead!**

Field name	**Value**	**Description**
Status code		Mandatory. Non-editable selection field. Empty by default. Provides a bunch of status codes, actually a part of HTTP response status

Field name	Value	Description
		codes. The selected status code is added to the HTTP response as HTTP response status.
	error	Error code.
	success	Success code.
keyValue1	*<KeyValue>*^{1}	Optional. Editable text field. Custom header property. First key-value pair that can be added to the HTTP response message header. Supports: • Properties (page 392), e.g. '$[MY_VALUE]' or 'Doctype= $[DOCTYPE]'. • System variables (page 392), e.g. '${MY_VALUE}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
keyValue2	*<KeyValue>*^{1}	Optional. Editable text field. Second key-value pair that can be added to the HTTP response message header. Refer to *Key=Value1* for more details.
keyValue3	*<KeyValue>*^{1}	Optional. Editable text field. Third key-value pair that can be added to the HTTP response message header. Refer to *Key=Value1* for more details.
logmessage	*<Text>*^{2}	Optional. Refer to Common Fields (page 3).

1) Key-value pair with notation: MyKey=MyValue.
2) Arbitrary text.

1.15 JMS Activities

This section describes JMS message related activities.

1.15.1 acJMSListQueue

Caption: JMSListQueue.

Throws Fault=acJMSListQueue.

The *acJMSListQueue* activity is a file activity that lists all message headers from the selected JMS destination and returns a FILELIST object (CSV file) that can be used as direct input to the *scForEach* (page 10) activity providing a loop element. Uses the *listQueue* operation from the JMS *Client Adapter* component (refer to chapter *1.21.3 listQueue (Listing Queue)* of the *BIS6 Base Components* manual or to chapter *8.5 listQueue (Listing Queue)* of the *JMS Client Adapter* manual *(link)* for more details).

Field name	Value	Description
Destination	*<Ref>*^{1}	Mandatory. Editable selection field. Master data JMS Client address. Represents a JMS destination. The *Receiver* supports: • Properties (page 392), e.g. '$[JMS_ADDRESS]'. • System variables (page 392), e.g. '${JMS_ADDRESS}'.
Connection	*<Ref>*^{1}	Mandatory. Editable selection field. Master data JMS Client connection. Holds connection settings using a specific JMS provider. The *Sender* supports:

Field name	Value	Description
		• Properties (page 392), e.g. '$[JMS_CONNECTION]'. • System variables (page 392), e.g. '${JMS_CONNECTION}'.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.

Additional configuration:

System variable	Description
MFT-FILELISTCREATOR	Name of mapping for conversion of list returned by the *JMS Client Adapter* into FILELIST CSV format. ⚠ Changing this mapping affects all list activities with FILELIST format!

Preconditions:

- JMS Client adapter started.
- BIC Client adapter started.
- Mapping from ${MFT-FILELISTCREATOR} system variable available.

Result:

- New process attachment: CSV file with JMS queue entries.

Sets the following properties:

- LIST_COUNT := number of JMS queue entries.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid *Destination*	Error. Throws acForwardingJMS fault.
Invalid *Connection*	Error. Throws acForwardingJMS fault.
Wrong configuration of *Destination*	Error. Throws acForwardingJMS fault.
Invalid property in *Destination* or *Connection*	Error. Throws acForwardingJMS fault.
Invalid system variable in *Destination* or *Connection*	Error. Throws acForwardingJMS fault.

Policy: none.

1.15.2 acJMSPickFile

Caption: JMS send.

Throws Fault=acJMSPickFile.

The *acJMSPickFile* activity is an activity that picks a message from a JMS queue. Uses the *pollQueue* and *getMessage* operations from the JMS *Client Adapter* component (refer to chapter *1.21.5 pollQueue (Polling Queue) (link)* resp. chapter *1.21.1 getMessage (Getting Message) (link)*of the *BIS6 Base Components* manual or to chapter *8.3 pollQueue (Polling Queue) (link)* resp. chapter *8.7 getMessage (Getting Message) (link)*of the *JMS Client Adapter* manual *(link)* for more details).

Mode of operation:
The *acJMSPickFile* activity picks a message from the JMS queue specified in the *Receiver* address, either the next one just by pollingt the queue or a message specified by its ID. If the message has been picked successfully an MFT-JMSEvent is triggered (refer to chapter *MFT-JMSEvent* in the *MFT Standard Solution - Adapters* manual for more details). This event gets the message that has been picked from the JMS queue: the message body is stored to the process attachment, the message properties are transferred. Be aware that this event gets a completely new track ID that is independent from the track ID of the process that contains this *acJMSPickFile* activity.

Field name	Value	Description
Receiver	*<Ref>*[1]	Mandatory. Not editable selection field. Master data JMS Client address. Specifies a JMS destination i.e. the JMS queue holding the messages to pick.
Connection	*<Ref>*[1]	Mandatory. Not editable selection field. Master data JMS Client connection. All settings to establish a connection using a specific JMS provider.
Get by ID		Provides 2 modes for picking a message.
	True	Default. Specify the message to pick by its ID. Makes *Message ID* visible. Uses the *getMessage* operation of the JMS Client Adapter.
	False	No specific file defined. Just pick the next message that is available. Uses the *pollQueue* operation of the JMS Client Adapter.
Message ID	*<MsgID>*[2]	Optional. Editable text field. Visible when *Get by ID* is switched on. Wildcards, properties, system variables and XPath expressions are not supported.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1). Do not mix up with the queue from the *Receiver* that holds the JMS messages. This *Queue* performs JMS adapter invocation only.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) JMS message ID, format 'ID:4647c094-532d-11e6-819b-90b11c91b18f'.
3) Arbitrary text.

Preconditions:

- JMS Client adapter started.

Result:

- New *MFT-JMSEvent* for the picked message.
- Current process attachment: not changed.

Sets the following properties:

- MESSAGE_COUNT := 0 or 1 in case of success.

Possible cases of failure:

Possible case of failure	Success or error behavior
JMS adapter not started	Error. Throws acJMSPickFile fault.
Wrong *Receiver* configuration	Error. Throws acJMSPickFile fault.
Wrong Connection configuration	Error. Throws acJMSPickFile fault.
Message with *Message ID* does not exist	Success. No *MFT-JMSEvent* triggered.

Policy: none

1.15.3 mdJMS

Caption: JMS send.

Throws Fault=acForwardingJMS.

The *acForwardingJMS* activity is a single file activity that sends the current process attachment using a variety of message types (see below). In addition, the JMS header properties may be set. Uses the *sendMessage* operation from the JMS *Client Adapter* component (refer to chapter *1.21.9 sendMessage (Sending Message) (link)* of the *BIS6 Base Components* manual or to chapter *8.1 sendMessage (Sending Message) (link)* of the *JMS Client Adapter* manual *(link)* for more details).

Field name	Value	Description
Receiver	*<Ref>*[1]	Mandatory. Not editable selection field. Master data JMS Client address. The *Receiver* supports: • Properties (page 392), e.g. '$[JMS_RECEIVER]'. • System variables (page 392), e.g. '${JMS_RECEIVER}'.
Connection	*<Ref>*[1]	Mandatory. Not editable selection field. Master data JMS Client connection. The *Sender* supports: • Properties (page 392), e.g. '$[JMS_CONNECTION]'. • System variables (page 392), e.g. '${JMS_CONNECTION}'.
JMS properties (key=value;key=value;...)	*<KVList>*[2]	Optional. Editable text field. Empty by default. List of properties that are added to the message. Maximum number of properties is 2000.The keys need to be valid Java identifiers. The *JMS properties (key=value;key=value;...)* supports: • Properties (page 392) for the value parts only, e.g. '$[VALUE]'. • System variables (page 392) for the value parts only, e.g. '${VALUE}'. • XPath expressions (page 392) for the value parts only, e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or

Field name	Value	Description
		$(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). - Maximum number of properties is 2000. Additional properties are ignored. ⚠ It is recommended to use property keys that do not start with 'JMS'. The *mdJMS* activity fails for most of these keys.
Message type		Mandatory. Not editable selection list. Provides several message types (Please refer to the topic **Message Types** below):
	BYTES	The message body contains an uninterrupted stream of bytes. Text attachments can be used since they are java.io.ByteArrayInputStream objects.
	MAP	The message body contains a set of name/value pairs, no order defined. The process attachment is expected to contain a serialized 'Map' object.
	MESSAGE	Default. Message without message body. Does not need the process attachment.
	OBJECT	The message body contains a serialized Java object. The process attachment is expected to contain a serializable object.
	STREAM	The message body contains a stream of Java primitive type which is read and written sequentially.
	Text	The message body contains a Java string, for instance an XML message. The process attachment is expected to contain text.
	XML	Special message for the Sonic MQ communication only.
Delivery mode		Mandatory. Non-editable selection field. Delivery mode for messages.
	<default>	Default. The JMS adapter's default value. Actually corresponding to PERSISTENT mode.
	NON_PERSISTENT	Non-persistent mode. Messages are not stored.
	PERSISTENT	Persistent mode. Messages are stored.
MQ data type		Optional. Editable selection field. Default value empty. Provides 2 predefined types. Sets the message type according to WebSphere native MQ notation. Properties, system variables and XPath expressions are not supported.
	MQIIH	WebSphereMQ message header.
	MQRFH2	WebSphereMQ message header.
	NONE	No message header.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Semicolon separated key-value list.
3) Arbitrary text.

Preconditions:

- JMS Client adapter started.
- Available process attachment for all message types except 'MESSAGE'. Attachment format depending on selected message type.

Result:

- Current process attachment: not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := 'Sent'.
- TARGET_NAME := Name of used JMS queue.
- MESSAGE_ID := JMS message ID, for instance 'ID:398ef0ee-4445-11e6-803c-90b11c91b18f'.

> This message ID can be used in the *acJMSPickFile* (page 252) activity in order to pick a certain message from the queue.

- DIRECTION := 'OUT'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid *Receiver*	Error. Throws acForwardingJMS fault.
Invalid *Connection*	Error. Throws acForwardingJMS fault.
Wrong configuration of *Receiver*	Error. Throws acForwardingJMS fault.
Key in *JMS properties* starting with 'JMS'	Error. Throws acForwardingJMS fault.
XML message type for provider other than SonicMQ	Error. Throws acForwardingJMS fault.
Attachment type does not match type expected by selected message type	Error. Throws acForwardingJMS fault.

Policies:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/JMS ALLOWANCE/SYSTEM-EXECUTE-SCRIPT

1.15.4 Message Types

Supported message types:

Type	Description
Message	This represents a message without a message body.
Stream	This is a message whose body contains a stream of Java primitive type. It is written and read sequentially.
Map	This is a message with the body containing a set of name/value pairs. The order of the entries is not defined.
Text	This is a message whose body contains a Java string; such as a XML message.
Object	This is a message whose body contains a serialized Java object.
Bytes	This is a message whose body contains a stream of uninterrupted bytes.
XML	This is a special message for the communication only with the Sonic MQ.

1.15.5 acJMSReply

Caption: JMSReply.

Throws Fault=acJMSReply.

The *acJMSReply* activity is a single file activity that sends the current process attachment as part of a reply to an incoming JMS request. Uses the *sendMessage* operation from the JMS *Client Adapter* component (refer to chapter *1.21.9 sendMessage (Sending Message) (link)* of the *BIS6 Base Components* manual or to chapter *8.1 sendMessage (Sending Message) (link)* of the *JMS Client Adapter* manual *(link)* for more details).

Mode of operation:
The *acJMSReply* activity has to be used as a part of the MFT-JMSEvent process, i.e. the Criteria *From adapter* field has to set to 'JMS event' (internal value is 'JMSCLI').The MFT-JMSEvent sets the $[MESSAGE_ID], $[CORRELATION_ID], and $[JMS_REPLY_DESTINATION] properties which are needed by the *acJMSReply* activity.

Field name	Value	Description
Correlation ID mode		Mandatory. Not editable selection field. Several modes of determining the correlation ID for the send operation.
	Set CorrelationId	Enter the correlation ID manually. Makes *CorrelationID* visible.
	Use (only) the received CorrelationId	Default. Use the correlation ID from $[CORRELATION_ID] property as correlation ID.
	Use (only) the received MessageId	Use the message ID from $[MESSAGE_ID] property as correlation ID.

Field name	Value	Description
	Use the CorrelationId, if unavailable fallback to the MessageId	Use the correlation ID from $[CORRELATION_ID] property if available. Otherwise use the message ID from $[MESSAGE_ID] property as correlation ID.
CorrelationID	<CorrID>	Mandatory. Editable text field. Correlation ID for the send operation. Supports: • Properties (page 392), e.g. '$[MY_CORRID]'. • System variables (page 392), e.g. '${MY_CORRID}'. Visible if *Correlation ID mode* is set to 'Set CorrelationId'.
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.

Preconditions:

- JMS Client adapter started.
- Available process attachment.
- Properties $[CORRELATION_ID] and/or $[MESSAGE_ID] depending on selected *CorrelationID mode*, generally set by received *MFT_JMSEvent*.
- Property $[JMS_REPLY_DESTINATION], generally set by received *MFT_JMSEvent*.

Result:

- Current process attachment, not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid system variable in *CorrelationID*	Error. Throws acJMSReply fault.

Policies: none.

1.16 SLMP Activities

This section describes SLMP related activities.

1.16.1 acSLMPQueueMessage

Caption: SLMP queue message

Throws Fault=acSLMPQueueMessage.

The *acSLMPQueueMessage* activity is an activity that wraps the current process attachment into an SLMP message and puts this message into a queue. Uses the *composeMessage* operation from the *SLMP Client* component and the *waitForPolling* operation from the *SLMP Controller* component (refer to chapter *1.39.1 composeMessage (Creates Message)* resp. chapter *1.40.4 waitForPolling (Sending Messages to the Queue)* of the *BIS6 Base Components* manual or to chapter *8.1.2 composeMessage (Creates Message)* resp. chapter *8.2.1 waitForPolling (Sending Messages to the Queue)* of the *SLMP Adapter* manual for more details).

Mode of operation:
ComposeMessage creates an SLMP message containing the current process attachment (using MIME content type application/octet-stream) and all required MIME headers. This SLMP message is sent to the direct mode queue as entered in *Queue*. It can be polled from the queue later on within the time span specified by *Validity*. If the SLMP message has not been polled when the *Validity* time span is expired the *acSLMPQueueMessage* activity runs on error.

Field name	Value	Description
Own ID	*<Ref>*[1]	Mandatory. Editable selection field. Master data SLMP own addresses and bidirectional addresses. Properties, system variables and XPath expressions are not supported.
Receiver ID	*<Ref>*[1]	Mandatory. Editable selection field. Master data SLMP partner addresses and bidirectional addresses. Supports: • System variables (page 392), ${FILENAME}'. • Properties (page 392), e.g. '$[FILENAME]'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Subject	*<Text>*[2]	Optional. Editable text field. Default value 'SLMP'. Message description. Can be used for (customer-specific) filtering. Supports: • System variables (page 392), ${SUBJECT}'. • Properties (page 392), e.g. '$[SUBJECT]'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
File name		Mandatory. Editable text field. File name used in MIME header. Supports: • System variables (page 392), ${FILENAME}'. • Properties (page 392), e.g. '$[FILENAME]'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).

Field name	Value	Description
Validity	*<Int>*[3]	Optional. Editable text field. Default value 3600 (seconds). The amount of time the created message is available for polling in the queue specified in *Queue*. Supports: • System variables (page 392), ${VALIDITY}'. • Properties (page 392), e.g. '$[VALIDITY]'.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.
3) Positive Integer number.

Preconditions:

- SLMP Client adapter started.
- Available process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- SLMP_QUEUE := *Queue* value.

Possible cases of failure:

Possible case of failure	Success or error behavior
SLMP Client adapter not started	Error. Throws acSLMPQueueMessage fault.
Invalid characters in validity	Error. Throws acSLMPQueueMessage fault.
Negative or zero validity	Error. Throws acSLMPQueueMessage fault.
SLMP message not polled within specified validity	Error. Throws acSLMPQueueMessage fault.

Policy: none.

1.16.2 mdSLMP

Caption: SEE Link send.

Throws Fault=acForwardingSLMP.

The *mdSLMP* activity is a single file activity that sends the current process attachment via SLMP. Uses the *sendMessage* operation from the *SLMP Client* component (refer to chapter *1.39.5 sendMessage (Sending Message)* of the *BIS6 Base Components* manual or to chapter *8.1.1 sendMessage (Sending Message)* of the *SLMP Adapter* manual for more details).

Mode of operation:

Creates and sends an SLMP message containing the current process attachment as payload to the specified receiver. The response contains a synchronous MDN report and specific details for the report of the sent SLMP message.

Field name	Value	Description
Own ID	*<Ref>*[1]	Mandatory. Editable selection field. Master data SLMP Own Address or Bidirectional Address. Properties and system variables are not supported.
Receiver ID	*<Ref>*[1]	Optional. Empty by default. Editable selection field. Master data SLMP Partner Address or Bidirectional Address. Supports: • Properties (page 392), e.g. '$[SLMP_PARTNER]'. • System variables (page 392), e.g. '${SLMP_PARTNER}'. The value is stored to the 'TO_NAME' property.
Subject	*<Text>*[2]	Optional. Default value 'SLMP'. Editable text field. Transmission description usable for (customer-specific) filtering. Supports: • Properties (page 392), e.g. '$[SUBJECT]'. • System variables (page 392), e.g. '${SUBJECT}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). The value is stored to the 'TO_SUBJECT' property.
File name		Mandatory. Default value 'SLMP'. Editable text field. File name used for the payload to be sent. Supports: • Properties (page 392), e.g. '$[FILE_NAME]'. • System variables (page 392), e.g. '${FILE_NAME}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). The value is stored to the 'TARGET_NAME' property.
Add parameters	True	Makes *Parameters to add* visible.
	False	Default.
Parameters to add (KEY=VALUE;PREFIX=SLMP_;...)	*<KVList>*[3]	Mandatory. Editable text field. Semicolon separated key-value list. Additional parameters sent with SLMP message. Supports: • Properties (page 392), e.g. '$[PARAM_LIST]' OR '$[KV1];$[KV2];$[KV1]'. • System variables (page 392), e.g. '${PARAM_LIST}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).

Field name	Value	Description
		Vsible when *Add parameters* is switched on.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Arbitrary text.
3) Semicolon separated key-value list.

Preconditions:

- SLMP Client adapter started.
- Available process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- COMMUNICATION_MESSAGE_ID := Message ID.
- TARGET_NAME := File name.
- TO_SUBJECT := Subject.
- TO_NAME := Receiver ID.
- TO_UUID := UUID of receiver.
- TO_ID := SLMPID of receiver.
- TO_ADAPTER := 'SLMPClient.sendMessage'.
- DIRECTION := 'OUT'.
- STATUS := 'SUCCESS_NOTIFY'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Own ID invalid, not existing or property	Error. Throws acForwardingSLMP fault.
Receiver ID invalid or not existing	Error. Throws acForwardingSLMP fault.
Receiver ID with non-existing property	Error. Throws acForwardingSLMP fault.
Receiver ID with non-existing system variable	Error. Throws acForwardingSLMP fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/SLMP

1.17 ICAP Activities

This sections describe activities for ICAP support.

⚠ Beta version only!

1.17.1 acICAP

Caption: ICAP adapter.

Throws Fault=acICAP.

⚠ Beta version only!

The *acICAP* activity is a single file activity that sends a request to a specified ICAP server. Uses the *Send* operation from the *ICAP Adapter* component (refer to chapter *1.20.1 Send (link)* of the *BIS6 Base Components* manual or to chapter *8.1 Send (link)* of the *ICAP Adapter* manual *(link)* for more details).

Field name	Value	Description
Process mode		Mandatory. Non editable selection mode. Supported ICAP operation modes.
	REQMOD	Default. Send HTTP request for pre-processing.
	RESPMOD	Send HTTP response for post-processing.
	OPTIONS	Request information for specific service on ICAP server. ⚠ Passing ICAP server information is not implemented.
Host	*<host>*[1]	Mandatory. Editable text field.Empty by default. The host name or IP address of the ICAP server.
Port	*<port>*[2]	Mandatory. Editable text field.Empty by default. The port of the ICAP server .
Service name		Mandatory. Editable text field.Empty by default. Name of remote service on ICAP server.
Proxy mode		Mandatory. Non editable selection mode. Provides supported proxy modes.
	NONE	Default. Use no proxy.
	SPECIFIC	Use simple proxy record.
	GROUP	Use group proxy record.
Proxy alias		Optional. Empty by default. Editable text field. Proxy record ID. The proxy type has to match *Proxy mode*.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Host name of IP address.
2) Valid port.
3) Arbitrary text.

Preconditions:

- ICAP client adapter started.
- Available process attachment.

Result:

- New process attachment if changed by ICAP server.

Sets the following properties:

- ICAP_CHANGED_INFO := Change information from ICAP server.

Possible cases of failure:

Possible case of failure	Success or error behavior
ICAP client adapter not started	Error. Throws acICAP fault.
No connection to ICAP server	Error. Throws acICAP fault.

Policy: none.

1.18 Database Activities

This section describes database related activities which enable reading data from a table as well as inserting, modifying and deleting data in a table.

1.18.1 acWriteSQL

Caption: Write SQL data.

Throws Fault=acWriteSQL.

The *acWriteSQL* activity is an activity that inserts, updates or deletes data in a database table. Uses the *writ eSQL* operation from the *DB Client Adapter* component (refer to chapter *1.4.3 writeSQL (adds, updates and removes data)* of the *BIS6 Base Components* manual or to chapter *8.2 writeSQL (adds, updates and removes data)* of the *Database Adapter* manual for more details).

Fieldname	Value	Description
Data source	*<Ref>*[1]	Mandatory. Non-editable selection field. Master data data source either XA or JDBC.
SQL operation		Mandatory. Non-editable selection field. Provides supported insert, update and delete operations. If you need more complex operations use BIC or a java script.
	INSERT	Default. Inserts new records into the database table using the data from the current process attachment in CSV or XML format.
	INSERT NO INPUT DATA	Inserts a new record into the database table using the data from the *Key value list*.
	UPDATE	Updates already existing records in the database table using the data from the current process attachment in CSV or XML format.
	DELETE	Deletes existing records from the table contained in the attachment in a specific form. Needs either CSV, XML or seeburger rowset as Input attachement. Otherwise the activity will fail.
	ADD BLOB	Adds the current process attachment as a binary image/blob into the specified column of one or more already existing records.

Fieldname	Value	Description
	UPDATE COLUMN	Updates a value in a specified column of one or more already existing records.

SQL operation: INSERT

The INSERT operation inserts the content of the current process attachment into the specified database table.

Fieldname	Value	Description
Table name	*<SQLTable>*[1]	Mandatory. Editable text field. The name of the database table where the data from the current process attachment are to be inserted.. The *Table name* supports: • System variables (page 392), e.g. '${TABLE_TABLE}'. • Properties (page 392), e.g. '$[TABLE_TABLE]'. • XPath expressions (page 392), e.g. $(upper-case('ttable')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Input format		Mandatory. Non-editable selection field. Provides supported formats for the input attachment.
	CSV	Default. CSV format with header line. Uses column delimiter semicolon ';', row delimiter new line '\n', and mask values double quote '&quot;'. Uses file encoding 'US-ASCII'.
	XML	XML format 'SeeburgerRowSet'. Uses file encoding UTF-8'.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).
Queue	*<Ref>*[3]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).

1) Valid name of existing database table.
2) Arbitrary text.
3) Reference to master data.

Preconditions:

- DB Client adapter started.
- Available process attachment: matching *Input format*, either CSV or XML.

Result:

- Current process attachment: not changed.

Sets the following properties:

- ROW_COUNT := number of written database records.
- STATUS := 'SUCCESS'.
- RESULT := 'SQL executed'.

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
Invalid table name		Error. Throws acReadSQL fault.
Empty process attachment		Error. Throws acReadSQL fault.
Process attachment with data format not matching *Input format*		Error. Throws acReadSQL fault.
Process attachment with header line only	CSV	Success. No data inserted.

Policy: none.

SQL operation: INSERT NO INPUT DATA

The INSERT NO INPUT DATA operation inserts the content of the *Key value list* into the specified database table.

Fieldname	Value	Description
Table name	*<SQLTable>*[1]	Mandatory. Editable text field. The name of the database table. The *Table name* supports: • System variables (page 392), e.g. '${TABLE_TABLE}'. • Properties (page 392), e.g. '$[TABLE_TABLE]. • XPath expressions (page 392), e.g. $(upper-case('ttable')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Key value list	*<KVlist>*[2]	Mandatory. Editable text field. List of column names and assigned values to be entered into the specified database table. String values must be surrounded by double quotes or single quotes, e.g. cColumn1=1;cColumn2='Wayne', or cColumn1=1;cColumn2="Wayne". Maximum number of entries is 20. The *Key value list* supports: • System variables (page 392), e.g. '${COLUMN1}="${NUMBER1}"'. • Properties (page 392), e.g. '$[COLUMN1]="$[NUMBER1]"'. • XPath expressions (page 392), e.g. $(upper-case('ttable')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).
Queue	*<Ref>*[4]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).

1) Valid name of existing database table.
2) Semicolon separated key-value list with existing columns of the specified database table and assigned values matching the column types.
3) Arbitrary text.
4) Reference to master data.

Preconditions:

- DB Client adapter started.

Result:

- Current process attachment: not changed.

Sets the following properties:

- ROW_COUNT := number of written database records.
- sStatus := 'SUCCESS'.
- sResult := 'SQL executed'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid table name	Error. Throws acReadSQL fault.
Syntax error in key-value list	Error. Throws acReadSQL fault.
Invalid column in key-value list	Error. Throws acReadSQL fault.

Policy: none.

SQL operation: UPDATE

The UPDATE operation updates already existing records in the specified database table with the content of the current process attachment in CSV or XML format.

Fieldname	Value	Description
Table name	*<SQLTable>*[1]	Mandatory. Editable text field. The name of the database table. The *Table name* supports: • System variables (page 392), e.g. '$[TABLE_NAME]'. • Properties (page 392), e.g. '$[TABLE_NAME]'.
Condition	*<SQLCondition>*[2]	Mandatory. Editable text field. Condition for SQL Where clause Specifies the records of the database table which are to be updated. If *Condition* is left empty, all records in the table are updated. The *Condition* supports: • System variables (page 392), e.g. '${SQL_CONDITION}'. • Properties (page 392), e.g. '$[SQL_CONDITION]'. • XPath expressions (page 392), e.g. $(upper-case('ccolumn')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Please note that this Gui field is optional and allows to store the activity without any condition entered. **Be aware** that this activity is stored and runs successfully and will **overwrite the specified columns of all records of the specified database table**.
Input format		Mandatory. Non-editable selection field. Provides supported formats for the input attachment. ⚠ It is recommended to have only 1 record in the input attachment. Only the last record is applied to the specified database table, i.e. its predecessors are being overwritten.

Fieldname	Value	Description
	CSV	Default. CSV format with header line. Uses column delimiter semicolon ';', row delimiter new line '\n', and mask values double quote '&quot;'. Uses file encoding 'US-ASCII'.
	XML	XML format 'SeeburgerRowSet'. Uses file encoding UTF-8'.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).
Queue	*<Ref>*[4]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).

1) Valid name of existing database table.
2) Valid SQL condition to be placed in WHERE clause.
3) Arbitrary text.
4) Reference to master data.

Preconditions:

- DB Client adapter started.
- Available process attachment: matching *Input format*, either CSV or XML.

Result:

- Current process attachment: not changed.

Sets the following properties:

- ROW_COUNT := number of written database records.
- sStatus := 'SUCCESS'.
- sResult := 'SQL executed'.

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
Invalid table name		Error. Throws acReadSQL fault.
Missing condition		Success. All entries of the specified database table are updated.
Invalid condition (invalid column name, wrong value type)		Error. Throws acReadSQL fault.
Empty process attachment		Error. Throws acReadSQL fault.
Process attachment with data format not matching *Input format*		Error. Throws acReadSQL fault.
Process attachment with header line only	CSV	Error. Throws acReadSQL fault.

Policy: none.

SQL operation: DELETE

The DELETE operation deletes existing records from the specified database table.

Fieldname	Value	Description
Table name	*<SQLTable>*[1]	Mandatory. Editable text field. The name of the database table. The *Table name* supports: • System variables (page 392), e.g. '${TABLE_TABLE}'. • Properties (page 392), e.g. '$[TABLE_TABLE]'. • XPath expressions (page 392), e.g. $(upper-case('ttable')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Condition	*<SQLCondition>*[2]	Optional. Editable text field. Condition for SQL Where clause. Specifies the records of the database table which are to be deleted. If *Condition* is left empty, all records in the table are deleted. The *Condition* supports: • System variables (page 392), e.g. '${SQL_CONDITION}'. • Properties (page 392), e.g. '$[SQL_CONDITION]'. • XPath expressions (page 392), e.g. $(upper-case('ccolumn')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Please note that this Gui field is optional and allows to store the activity without any condition entered. **Be aware** that this activity is stored and runs successfully and will **delete the entire content of the specified database table**.
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).
Queue	*<Ref>*[4]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).

1) Valid name of existing database table.
2) Valid SQL condition to be placed in WHERE clause.
3) Arbitrary text.
4) Reference to master data.

Preconditions:

- DB Client adapter started.

Result:

- Current process attachment: not accessed and not changed.

Sets the following properties:

- ROW_COUNT := number of written database records.
- sStatus := 'SUCCESS'.
- sResult := 'SQL executed'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid table name	Error. Throws acReadSQL fault.
Missing condition	Success. All entries of the specified database table are deleted.
Invalid condition (invalid column name, wrong value type)	Error. Throws acReadSQL fault.

Policy: none.

SQL operation: ADD BLOB

The ADD BLOB operation adds the current process attachment as a binary image/blob into the specified column of already existing records.

Fieldname	Value	Description
Table name	*<SQLTable>*[1]	Mandatory. Editable text field. The name of the database table. The *Table name* supports: • System variables (page 392), e.g. '${TABLE_TABLE}'. • Properties (page 392), e.g. '$[TABLE_TABLE]'. • XPath expressions (page 392), e.g. $(upper-case('ttable')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Column name	*<SQLColumn>*[2]	Mandatory. Editable text field. The name of the database table column that contains blobs. The *Column name* supports: • System variables (page 392), e.g. '${COLUMN_NAME}'. • Properties (page 392), e.g. '$[TABLE_TABLE]'. • XPath expressions (page 392), e.g. $(upper-case('columname')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Condition	*<SQLCondition>*[3]	Optional. Editable text field. Condition for SQL Where clause. Specifies the records of the database table to whom the blob is to be added. If *Condition* is left empty the blob is added to all records. The *Condition* supports: • System variables (page 392), e.g. '${SQL_CONDITION}'. • Properties (page 392), e.g. '$[SQL_CONDITION]'. • XPath expressions (page 392), e.g. $(upper-case('ccolumn')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Please note that this Gui field is optional and allows to store the activity without any condition entered. **Be aware** that this activity is stored and runs successfully and will **overwrite the specified columns of all records of the specified database table**.
Log message	<Text>4	Optional. Refer to Common Fields (page 3).
Queue	*<Ref>*[5]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).

1) Valid name of existing database table.
2) Valid name of existing column in specified database table.
3) Valid SQL condition to be placed in WHERE clause.
4) Arbitrary text.
5) Reference to master data.

Preconditions:

- DB Client adapter started.

Result:

- Current process attachment: not accessed and not changed.

Sets the following properties:

- ROW_COUNT := number of written database records.
- sStatus := 'SUCCESS'.
- sResult := 'SQL executed'

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid table name	Error. Throws acReadSQL fault.
Invalid column name	Error. Throws acReadSQL fault.
Missing condition	Success. All values in the specified column are overwritten.
Invalid condition (invalid column name, wrong value type)	Error. Throws acReadSQL fault.

Policy: none.

SQL operation: UPDATE COLUMN

The UPDATE COLUMN operation updates a value in a specified column of already existing records.

Fieldname	Value	Description
Table name	*<SQLTable>*[1]	Mandatory. Editable text field. The name of the database table. The *Table name* supports: • System variables (page 392), e.g. '${TABLE_TABLE}'. • Properties (page 392), e.g. '$[TABLE_TABLE]'. • XPath expressions (page 392), e.g. $(upper-case('ttable')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Column name	<*SQLColumn*>[2]	Mandatory. Editable text field. The name of the target column whose value is to be changed for all records specified by *Condition*. The *Column name* supports: • System variables (page 392), e.g. '${COLUMN_NAME}'. • Properties (page 392), e.g. '$[TABLE_TABLE]'.

Fieldname	Value	Description
		• XPath expressions (page 392), e.g. $(upper-case('columname')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000'))
New value	*<ColValue>*[3]	Mandatory. Editable text field. The new value for the column specified by *Column name*. The *New value* supports: • System variables (page 392), e.g. '${VALUE}'. • Properties (page 392), e.g. '$[VALUE]'. • XPath expressions (page 392), e.g. $(upper-case('value')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000'))
Condition	*<SQLCondition>*[4]	Optional. Editable text field. Condition for SQL Where clause. Specifies the records of the database table where *New value* is to be entered into the column specified by *Column name*. If *Condition* is left empty *New value* is entered into all records' columns, The *Condition* supports: • System variables (page 392), e.g. '${SQL_CONDITION}'. • Properties (page 392), e.g. '$[SQL_CONDITION]'. • XPath expressions (page 392), e.g. $(upper-case('ccolumn')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). ⚠ Please note that this Gui field is optional and allows to store the activity without any condition entered. **Be aware** that this activity is stored and runs successfully and will **overwrite the specified columns of all records of the specified database table with the new value**.
Log message	*<Text>*[5]	Optional. Refer to Common Fields (page 3).
Queue	*<Ref>*[6]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).

1) Valid name of existing database table.
2) Valid name of existing column in specified database table.
3) Arbitrary value whose format has to match the type of the specified column.
4) Valid SQL condition to be placed in WHERE clause.
5) Arbitrary text.
6) Reference to master data.

Preconditions:

- DB Client adapter started.

Result:

- Current process attachment: not accessed and not changed..

Sets the following properties:

- ROW_COUNT := number of written database records.
- sStatus := 'SUCCESS'.
- sResult := 'SQL executed'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid table name	Error. Throws acReadSQL fault.
Invalid column name	Error. Throws acReadSQL fault.
Missing condition	Success. All values in the specified column are overwritten by the new value.
Invalid condition (invalid column name, wrong value type)	Error. Throws acReadSQL fault.

Policiy: none

1.18.2 acReadSQL

Caption: Read SQL data.

Throws Fault=acReadSQL.

The *acReadSQL* activity is an activity that reads data from a database table and puts the formatted result data into the process attachment. Uses the *readSQL* operation from the *DB Client Adapter* component (refer to chapter *1.4.2 readSQL (reads data)* of the *BIS6 Base Components* manual or to chapter *8.1 readSQL (reads data)* of the *Database Adapter* manual for more details).

Field name	Value	Description
Data source	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides available data sources.
SQL statement	<Select>[2]	Mandatory. Editable text field. Default value 'SELECT * FROM tTable WHERE cID='1''. Supports: • System variables (page 392), e.g. 'SELECT * FROM tWorkflow WHERE cID=$[MY_RECORD]'. • Properties (page 392), e.g. 'SELECT * FROM ${SQL_TABLE} WHERE cID=1'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Result format		Mandatory. Non-editable selection field. Provides supported formats for the result attachment.
	CSV	Default. CSV format with header line. Uses column delimiter semicolon ';', row delimiter new line '\n', and mask values double quote '&quot;'. Uses file encoding 'US-ASCII'.
	XML	XML format 'SeeburgerRowSet'. Uses file encoding UTF-8'.
Queue	*<Ref>*[1]	Mandatory. Not editable selection field. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Correct SELECT statement.
3) Arbitrary text.

Preconditions:

- DB Client adapter started.

Result:

- New process attachment: formatted result data of SQL query, if the Result format is set to 'CSV' or 'XML'.

Sets the following properties:

- ROW_COUNT := number of database records returned.
- STATUS := 'SUCCESS'.
- RESULT := 'SQL executed'.

Possible cases of failure:

Possible case of failure	Success or error behavior
System variable in *SQL statement* not found	Error. Throws acReadSQL fault.
The SQL query fails	Error. Throws acReadSQL fault.
SQL statement different from SELECT query, e.g. UPDATE, DELETE, etc.	Error. Throws acReadSQL fault.
No dataset found	Success. ROW_COUNT := 0.

Policiy: none

1.19 SAP Activities

This section describes SAP related activities.

1.19.1 acRFCClient

Caption: acRFCClient .

Throws Fault=acRFCClient.

The *acRFCClient* activity is a single file activity that executes an RFC function call on a SAP system using the XML function settings of the current process attachment. Uses the *executeRFC* operation of the RFC Client module (refer to chapter *executeRFC (link)* of the *BIS6* manual or to chapter *9.2 executeRFC (link)* of the *SAP RFC Client Adapter* manual for more details).

Mode of operation:
Execute an RFC function call on a SAP system. Hence RFC needs a specific schema, a conversion will have to prepare the attachment to suit the RFC function call before.

Field name	Value	Description
SAP address	*<Ref>*[1]	Mandatory. Editable selection field. Master data SAP addresses. Supports:

Field name	Value	Description
		• Properties (page 392), e.g. '$[SAP_ADDRESS]'. • System variables (page 392), e.g. '${SAP_ADDRESS}'.
Schema validation	True	Activates validation: RFC Client validates the process attachment (XML) with corresponding XSD schema.
	False	Default. No schema validation by RFC Client.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Arbitrary text.

Preconditions:

- SAP RFC Client adapter started.
- Available process attachment: XML format, function settings for the RFC call.

Result:

- New process attachment: XML response message, all fields of the called function.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid system variable in *SAP address*	Error. Throws acRFCClient fault.

Policies: none.

1.19.2 acSAPStatusReport

Caption: SAPStatusReport .

Throws Fault=acSAPStatusReport.

The *acSAPStatusReport* activity is an activity that sends status IDoc messages back to an R/3 system.

Uses the *executeBAPI* operation of the *BICNode* module or the *sendSAPStatusReport* or *sendSAPStatusReportSync* operations of the SAP Client adapter (refer to chapter *sendSAPStatusReport* and chapter *sendSAPStatusReportSync* of the *BIS6 Base Componenents* manual or to chapter *8.2.1 sendSAPStatusReport* and chapter *8.2.2 sendSAPStatusReportSync* of the *SAP IDoc Connector* manuall for more details) depending on the selected *Status report type*.

Field name	Value	Description
SAP status		Mandatory. Editable selection field. Provides supported SAP statuses. Default status is "12 Dispatch OK".

Field name	Value	Description
		Supports • Properties, e.g. '$[SAP_STATUS]'. • System variables, e.g. '${SAP_STATUS}'.
	10 Interchange handling OK	Interchange handling OK.
	09 Error during interchange handling	Error during interchange handling.
	05 Error during translation	Error during translation.
	06 Translation OK	Translation OK.
	11 Error during dispatch	Error during dispatch.
	12 Dispatch OK	Default. Dispatch OK.
	18 Triggering EDI subsystem	Triggering EDI subsystem OK.
	14 Interchange Acknowlegdement	Interchange Acknowlegdement positive.
	15 Interchange Acknowlegdement	Interchange Acknowlegdement negative.
	16 Functional Acknowlegdement	Functional Acknowlegdement positive.
	17 Functional Acknowlegdement	Functional Acknowlegdement negative.
	53 Application document posted	Application document posted.
	51 Application document not posted	Application document not posted.
	56 IDoc with errors added	IDoc with errors added
	64 IDoc ready to be transferred to application	64 IDoc ready to be transferred to application.
Status report type		Mandatory. Editable selection field. Provides supported status report types. System variables and properties are not supported.
	RFC (asynchron)	Default. Asynchronous transmission. Process does not wait for a response and continues with the next activity.

Field name	Value	Description
	RFC (synchron)	Synchronous transmission. Process waits for the response.
	BAPI report	Use a BAPI mapping that creates a BAPI report.
Archive key	*<Text>*[1]	Optional. Editable text field. Key of the external message archive (document ID in external system). Supports • Properties, e.g. '$[SAP_ARCKEY]'. • System variables, e.g. '${SAP_ARCKEY}'.
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.

Additional configuration:

System variable	Description
MFT-SAP-STATUS-REPORT-SYNC	Default value "FALSE".
MFT-BAPI-STATUS -MAPPING	Default value "SSC_BAPI_SET_STATUS". BAPI mapping name, needed when *Status report type* is set to 'BAPI report'.

Required properties:

Property	Description
SAP_PREPARED_REPORT_MESSAGE	Actual report message (payload).
SAP_IDOCNUMBER	Semicolon separated list of IDOC numbers. Maximum length 2000.
SAP_REPORTS_EXPECTED	'True': sends the status report. 'False': does not send the status report.
SAP_DESTINATION_ADDRESS_ID	SAP address ID, needed when *Status report type* is set to 'BAPI report'.

Preconditions:

- SAP IDoc Client Adapter started.
- Available process attachment: IDoc report.
- Available SAP address and BAPI mapping in case of *Status report type* is set to 'BAPI report'.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- SAP_STATUS := SAP status value (propertyBag).
- ARC_KEY := Archive key value (propertyBag).

Possible cases of failure:

Possible case of failure	Success or error behavior
System variable not exsting	Error. Throws acSAPStatusReport fault.

Policies: none.

1.19.3 acForwardingSAP

Caption: Forwarding SAP .

Throws Fault=acForwardingSAP.

The *acForwardingSAP* activity is a single file activity that sends the current process attachment (IDoc format) to an R/3 system. Uses the *SendIDocMD_v2* operation of the SAP IDoc Connector adapter (refer to chapter *SendIDocMD (link)* of the *BIS6 Base Components* or to chapter *8.1.1 SendIDocMD (link)* of the *SAP IDoc Connector* manual for more details).

Field name	Value	Description
SAP address	*<Ref>*[1]	Mandatory. Editable selection field. Master data SAP addresses. Supports: • Properties (page 392), e.g. '$[SAP_ADDRESS]'. • System variables (page 392), e.g. '${SAP_ADDRESS}'.
Abort on error	True/False	Refer to Common Fields.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields.
Log message	*<Text>*[2]	Optional. Refer to Common Fields.

1) Reference to master data record.
2) Arbitrary text.

Preconditions:

- Current process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- SAP_HOST := SAP host name.
- SAP_CLIENT := SAP client name.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid system variable in *SAP address*	Error. Throws acForwardingSAP fault.

Policies: none.

1.20 SEEFX Activities

This section describes SEEFX related activities which need a connection to a SEEFX server. Uses the SLMP Client adapter as well as the WebService Client adapter.

1.20.1 acSEEFXCreateDownloadPermit

Caption: SEEFXCreateDownloadPermit.

Throws Fault=acSEEFXCreateDownloadPermit.

The *acSEEFXCreateDownloadPermit* activity is a web service based activity that delivers a download link to a single file from a SEEFX account to all permit recipients in the list via email. Uses the *CreateDownloadPermit*, the *SendProtectedPermitMails*, and the *SendPermitMails* operations from the FileTransferService3 SEE FX web service (refer to chapter *1.2 FileTransferService3* of the *BIS FileExchange WebService API HowTo* manual for more details).

Mode of operation:
Creates a permit link (URL) to access the file for download based on given permit options. Sends a notification email with the download permit link to each of the permit recipients (i.e. mail type 'Link'). If desired, sends an additional notification email with the the downlaod password (i.e. mail type 'Password'). The content type of the notification emails is 'text/html' in each case.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to FileTransferService3Port web service.
Extended settings	True/False	Optional. Additional field group. Refer to Extended settings (page 304).
Full path to download file	*<FilePath>*[2]	Optional. Editable text field. Empty by default. Absolute path to a SEE FX file. Must follow the SEE FX Paths guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_FILE_PATH]'. • System variables (page 392), e.g. '${MY_FILE_PATH}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Wildcards are not supported.
Permit recipient list	*<EMailList>*[3]	Mandatory. Editable text field. Semicolon separated list of email addresses of permit recipients who will receive the notification emails with the permit link/s. Maximum number of list entries is 10. Additional entries are ignored. Supports: • Properties (page 392), e.g. '$[PERMIT_RECIPIENTS]'. • System variables (page 392), e.g. '${PERMIT_RECIPIENTS}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Mail body from bookmark	True	Get the mail body for the notification emails to the permit recipients from a bookmarked attachment.

Field name	Value	Description
		Makes *Mail body of download permit from bookmark* visible.
	False	Default. Get the mail body for the notification emails to the permit recipients directly from a text field. Makes *Mail body of download permit* or *Mail body of protected download permit* visible, and *Password protected download*.
Mail body of download permit from bookmark	*<BookmarkName>*[4]	Optional. Editable text field. Default value 'SEEFX_MAIL_BODY'. The name of a bookmarked attachment that contains the designated mail body for the notification emails to the permit recipients. Password protection is not supported. The content type is 'text/html'. The mail type is 'Link'. System variables and properties are not supported. The attachment content supports: • $P{links} placeholder: replaced with a list of permit links. I This placeholder is mandatory. If it is missing, the SEE FX mail body default template is used. Visible when *Mail body from bookmark* is switched on.
Mail body of download permit	*<Text>*[5]	Optional. Editable text field. Default value 'File download at $P{links} and pass $P{password} valid until %VALID_UNTIL% for %MAX_NUMBER_OF_DOWNLOADS% times available for your service!'. Mail body text for the notification emails to the permit recipients. If no text is entered, a default mail body text is used. Supports: • $P{links} placeholder: replaced with a list of permit links. I This placeholder is mandatory. If it is missing, the SEE FX mail body default template is used. • %VALID_UNTIL% placeholder: replaced with the date corresponding to the value from *Validation for days*. Date format is 'yyyy-MM-ddTHH:mm:ss'. • %MAX_NUMBER_OF_DOWNLOADS% placeholder: replaced with the value from *Maximum number of downloads*. • Properties (page 392), e.g. '$[MY_MAIL_BODY]'. • System variables (page 392), e.g. '${MY_MAIL_BODY}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Mail body from bookmark* is switched off and *Password protected download* is switched off.
Mail body of protected download permit	*<Text>*[5]	Optional. Editable text field. Default value 'File download at $P{links} and pass $P{password} valid until %VALID_UNTIL% for %MAX_NUMBER_OF_DOWNLOADS% times available for your service!'. Mail body text for the notification emails to the permit recipients. If no text is entered, a default mail body text is used. Supports: • $P{links} placeholder: replaced with a list of permit links. • $P{password} placeholder: replaced with the permit password.

Field name	Value	Description
		- %VALID_UNTIL% placeholder: replaced with the date corresponding to the value from *Validation for days*. Date format is 'yyyy-MM-ddTHH:mm:ss'. - %MAX_NUMBER_OF_DOWNLOADS% placeholder: replaced with the value from *Maximum number of downloads*. - Properties (page 392), e.g. '$[MY_MAIL_BODY]'. - System variables (page 392), e.g. '${MY_MAIL_BODY}'. - XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Mail body from bookmark* is switched off and *Password protected download* is switched on.
Mail subject	*<Text>*[5]	Optional. Editable text field. Empty by default. Subject for the notification emails to the permit recipients. Supports: - Properties (page 392), e.g. '$[MY_MAIL_SUBJECT]'. - System variables (page 392), e.g. '${MY_MAIL_SUBJECT}'. - XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Locale identifier	*<Locale>*[6]	Optional. Non-editable selection field. Provides supported locales. Locale identifier for the notification emails to the permit recipients.
Validation for days	*<Integer>*[7]	Mandatory. Default value 3. Editable number field. Number of days the permit remains valid. Minimum value 1, maximum value 30.
Maximum number of downloads	*<Integer>*[7]	Mandatory. Default value 5. Editable number field. Maximum number of allowed file downloads. Minimum value 1, maximum value 999.
Password protected download		Password protection configuration for non-bookmark mail bodies. Visible when *Mail body from bookmark* is switched off.
	True	Default. Enable password protection for the permit. Makes *Mail body of protected download permit* visible.
	False	Disable password protection for the permit. Makes *Mail body of download permit* visible.
Transfer password	*<Password>*[8]	Optional. Editable password field. Empty by default. Explicit password for permit. If left empty, the password is generated automatically if the *Password protected download* is switched on. System variables and properties are not supported.
Notify creator	True	Send a notification to the permit creator upon permit access by the permit recipients or not (false).
	False	Default. No notifications for the permit creator.
Subject	*<Text>*[5]	Optional. Editable text field. Empty by default. Subject reference for the permit. To be used as subject/reference in the notifications, in the form of: "Re: Subject". Supports: - Properties (page 392), e.g. '$[MY_SUBJECT]'.

Field name	Value	Description
		• System variables (page 392), e.g. '${MY_SUBJECT}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Abort on error	True/False	Mandatory. Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[5]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Absolute SEEFX file path.
3) Semicolon separated list of syntactically correct email addresses.
4) Arbitrary case-sensitive name except invalid characters ", &, <.
5) Arbitrary text.
6) Locale identifier. Format 'll_CC'. ll is a lowercase two-letter ISO-639 language code. CC is an uppercase two-letter ISO-3166 country code. E.g. en_US.
7) Positive Integer value.
8) Secure password.

Preconditions:

- FileTransferService3 web service installed.
- WebService client adapter started.

Result:

- Current process attachment: unused and not changed.

Sets the following properties:

- TARGET_PATH := Absolute path to folder of file to download.
- TARGET_NAME := Name of file to download.
- FULL_FILE_PATH_FOLDER := Absolute path of file to download.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
WebService client adapter not started		Error. Throws acSEEFXCreateDownloadPermit fault.
No connection to SEEFX server		Error. Throws acSEEFXCreateDownloadPermit fault.
Wrong/Invalid URL	Extended settings enabled	Error. Throws acSEEFXCreateDownloadPermit fault.
Wrong HTTP basic authentication	Extended settings enabled	Error. Throws acSEEFXCreateDownloadPermit fault.
Non-existing system variable in *U RL*	Extended settings enabled	Error. Throws acSEEFXCreateDownloadPermit fault.
Non-existing system variable in *User name*	Extended settings enabled	Error. Throws acSEEFXCreateDownloadPermit fault.
Non-existing bookmark for mail body	Mail body from bookmark enabled	Error. Throws acSEEFXCreateDownloadPermit fault.
Target SEE FX file and/or folder not existing		Error. Throws acSEEFXCreateDownloadPermit fault.

Policy: none.

1.20.2 acSEEFXDownloadFile

Caption: SEEFX download file (WebService).

Throws Fault=acSEEFXDownloadFile.

The *acSEEFXDownloadFile* activity is a web service based activity that downloads a single file from a SEEFX account. Uses the *DownloadFile* operation from the *FileTransferService3* SEEFX web service (refer to chapter *1.2* of the manual for more details).

Mode of operation:
Downloads a single file from a SEEFX account and returns it as current process attachment. Offers several options for file handling of the original file on the SEEFX account, e.g deleting, moving or doing nothing.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to FileTransferService3Port web service.
Extended settings	True/False	Optional. Additional field group. Default fields *Login with Token*, *URL* , *User name*, *Password* and *Login Token* are described in Extended settings with Token (page 305). Makes *Login with Token*, *URL*, *User name*, *Password* and *Login token* visible.

Field name	Value	Description
Source path	*<FilePath>*^{3}	Mandatory. Editable text field. Default value '/My Files/'. Absolute path to the SEE FX file to download. Must follow the SEE FX Path guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_FILE_PATH]'. • System variables (page 392), e.g. '${MY_FILE_PATH}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Operation after file is downloaded		Mandatory. Non-editable selection field. Provides supported file actions to be performed after download.
	Do nothing	Default. Does nothing. The downloaded file is kept. Internal value is 'DO_NOTHING'.
	Move file to another folder	Moves the downloaded file to another location. Internal value is 'REMOVE_FILE'. Makes *Move to folder path* visible.
	Move file to recycle bin	Moves the downloaded file to the recycle bin. Internal value is 'MOVE_TO_RECYCLE_BIN'.
	Remove file	Deletes the downloaded file immediately and permanently. Internal value is 'MOVE_FILE'.
Move to folder path	*<FilePath>*^{3}	Optional. Editable text field. Default value '/My Files/SENT/'. Supports: • Properties (page 392), e.g. '$[MY_FILE_PATH]'. • System variables (page 392), e.g. '${MY_FILE_PATH}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Operation after file is downloaded* is switched to 'Move file to another folder'.
Abort on error	True/False	Mandatory. Refer to Common Fields (page 3).
Queue	*<Ref>*^{1}	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*^{4}	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Session token from SEEFX server.
3) Absolute SEE FX file path.
4) Arbitrary text.

Preconditions:

- FileTransferService3 web service installed.
- WebService client adapter started.

Result:

- New process attachment: downloaded file.

Sets the following properties:

- SIZE := The size of the downloaded file.
- LAST_MODIFIED := The last-modified timestamp of the downloaded file.
- DESCRIPTION := The description of the downloaded file.
- CREATEDBY := The name of the user who created the downloaded file.
- CREATED := The creation timestamp of the downloaded file.
- TARGET_PATH := The path of the downloaded file.
- TARGET_NAME := The name of the downloaded file.
- OPERATION_AFTER_DOWNLOAD := The value from *Operation after file is downloaded*. Internal values are 'DO_NOTHING', 'REMOVE_FILE', 'MOVE_TO_RECYCLE_BIN', and 'MOVE_FILE'.

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
WebService client adapter not started		Error. Throws acSEEFXCreateDownloadPermit fault.
No connection to SEEFX server		Error. Throws acSEEFXCreateDownloadPermit fault.
Wrong/Invalid URL	*Extended settings* enabled	Error. Throws acSEEFXCreateDownloadPermit fault.
Wrong HTTP basic authentication	*Extended settings* enabled	Error. Throws acSEEFXCreateDownloadPermit fault.
Non-existing system variable in *U RL*	*Extended settings* enabled	Error. Throws acSEEFXCreateDownloadPermit fault.
Non-existing system variable in *User name*	*Extended settings* enabled	Error. Throws acSEEFXCreateDownloadPermit fault.
Non-existing SEEFX target file or folder (*Source path*)		Error. Throws acSEEFXCreateDownloadPermit fault.
Non-existing system variable in *Source path*		Error. Throws acSEEFXCreateDownloadPermit fault.
Non-existing property in *Source path*		Error. Throws acSEEFXCreateDownloadPermit fault.
Non-existing SEEFX folder *Move to folder path*	Selected *Move file to another folder*	Error. Throws acSEEFXCreateDownloadPermit fault.
Non-existing system variable in *Move to folder path*	Selected *Move file to another folder*	Error. Throws acSEEFXCreateDownloadPermit fault.
Non-existing property in *Move to folder path*	Selected *Move file to another folder*	Error. Throws acSEEFXCreateDownloadPermit fault.
Target file already existing when moving file	Selected *Move file to another folder*	Success. File is renamed.

Policy: none.

1.20.3 acSEEFXFilesSend

Caption: Send file to SEE FX.

Throws Fault=acSEEFXFilesSend.

The *acSEEFXFilesSend* activity is an activity, partially SLMP based and partially web service based, that allows sending the current process attachment to one or several SEEFX users. Uses the *sendFile* operation of the *SEE FX SLMP Client* adapter (refer to chapter *6.13 BIS Integeration with SLMP (link)* sub-chapter *6.13.4 Available BIS FileExchange operations* of the *BIS FileExchange Installation and Administration Guide* manual for more details) or the *SendAttached* operation of the *SendFileService* SEEFX web service (refer to chapter *3.1*

SendFileService (link) of the *BIS FileExchange WebService API HowTo* manual for more details) depending on the selected *Transport* adapter.

Field name	Value	Description
File name	*<FileName>*[1]	Mandatory. Editable text field. Empty by default. Supports: • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Subject	*<Text>*[2]	Mandatory. Editable text field. Empty by default. Subject of the send file operation. Supports: • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Text message	*<Text>*[2]	Mandatory. Editable text field. Empty by default. Message of the send file operation. Supports: • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Request download report	True	Sends a download notification to the BIS launching a MFT-SEEFXNotificationEvent.
	False	Default. No notification when a recipient downloads the sent file.
Expires after hours	<Integer>3	Optional. Editable number field. Default value 5 hours. Period of time in hours which the file is available to the recipient
Recipient	*<Ref>*[4]	Mandatory. Editable selection field. Master data BIS FileExchange address. Recipient can be a single SEE FX user or a semicolon separated list of recipients. Supports: • Semicolon separated list of recipients in case of 'SLMP Integration' selected in *Transport* field. Do not use this list format in case of WebService Integration. • Properties (page 392), e.g. '$[MY_FILE_NAME]'. • System variables (page 392), e.g. '${MY_FILE_NAME}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Transport		Mandatory. Non-editable selection field. Supported adapters for sending the file.

Field name	Value	Description
	via SLMP integration	Default. Use the SEEFX SLMP Client adapter. ⚠ The SEEFX SLMP interface operations will be deprecated in the future. It is recommended to use the WebService integration instead. Makes additional SLMP related fields visible (see below).
	via WebService integration	Use the SEEFX WebService Client adapter. Makes additional web service related fields visible (see below).
...	...	Fields depending on value selected in *Transport*. Refer to sections **Transport: SLMP integration** and **Transport: WebService integration** below for more details about involved fields.
Queue	*<Ref>*[4]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Mandatory. Refer to Common Fields (page 3).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Valid file name.
2) Arbitrary text.
3) Positive Integer number.
4) Reference to master data record.

Transport: SLMP integration

Uses the *sendFile* operation of the *SEEFX SLMP Client* adapter (refer to chapter *6.13 BIS Integration with SLMP (link)* sub-chapter *6.13.4 Available BIS FileExchange operations* of the *BIS FileExchange Installation and Administration Guide* manual for more details).

⚠ The SEEFX SLMP interface operations will be deprecated in the future. It is recommended to use the WebService integration instead.

Field name	Value	Description
Transport	via SLMP integration	Makes *SEEFX host* and *Sender* visible.
SEEFX host	*<Ref>*[1]	Mandatory. Non-editable selection field. Master data BIS FileExchange connection. Visible when *Transport* is set to 'via SLMP integration'.
Sender	*<Ref>*[1]	Mandatory. Non-editable selection field. Master data BIS FileExchange Personality address or Bidirectional address. Supports: • Properties (page 392), e.g. '$[MY_SENDER]'. • System variables (page 392), e.g. '${MY_SENDER}'. • XPath expressions (page 392), e.g. $(upper-case('sender')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Transport* is set to 'via SLMP integration'.

1) Reference to master data record.

Preconditions:

- SLMP Client adapter started.
- Available process attachment.

Result:

- Current process attachment: unchanged.

Sets the following properties:

- SEEFX_REPORT := Error / Success message.
- SEEFX_STATUS := '200' in case of success, '204' in case of failure or partial failure.
- SEEFX_TRANSFERID := The SEE FX transfer ID.
- STATUS := value of SEEFX_STATUS

Send file to SEE FX does not fail in case sender or recipient could not be found or are not visible to each other. The activity returns properties with status information instead.

Possible cases of failure:

Possible case of failure	Success or error behavior
No connection to SEEFX server	Error. Throws acSEEFXFilesSend fault.
Invalid recipient	Success. STATUS := '204'. LAST_STATUS := '204'. *Abort on error* does not work if switched on. In order to abort the process, *scConditionalStop* (page 17) or *scConditionalExit* (page 22) can be used with the condition '204'='$[SEEFX_STATUS]'. In order to handle this error case, *scConditionalFork* (page 15) or *scConditionalBranch* (page 14) can be used.
Invalid system variable in *Sender*	Error. Throws acSEEFXFilesSend fault.

Policy: None.

Transport: WebService integration

Uses the *SendAttached* operation of the *SendFileService* SEEFX web service (refer to chapter *3.1 SendFileService (link)* of the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
Recipient type		Mandatory. Editable selection field. Supported recipient type. Supports: • Properties (page 392), e.g. '$[MY_RECTYPE]'. • System variables (page 392), e.g. '${MY_RECTYPE}'. • XPath expressions (page 392), e.g. $(upper-case('sender')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')) Visible when *Transport* is set to 'via WebService integration'.

Field name	Value	Description
	Group	
	User	Default.
Transport	via WebService integration	Makes *Recipient type*, *WebService endpoint*, and *Extended settings* visible.
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to SendFileServicePort web services. Visible when *Transport* is set to 'via WebService integration'.
Extended settings	True	Use extended settings. Display *SENDER*, *Password*, and *URL* which overwrite the settings from the *WebService endpoint*. Makes *Sender* (below), *Password*, and *URL* visible. Visible when *Transport* is set to 'via WebService integration'.
	False	Default. No additional fields displayed.
Sender	*<Ref>*[1]	Mandatory. Non-editable selection field. Supports: • Properties (page 392), e.g. '$[MY_SENDER]'. • System variables (page 392), e.g. '${MY_SENDER}'. • XPath expressions (page 392), e.g. $(upper-case('sender')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Transport* is set to 'via WebService integration' and *Extended settings* is switched on.
Password	*<Password>*[2]	Mandatory. Editable password field. System variables and properties are not supported. Visible when *Transport* is set to 'via WebService integration' and *Extended settings* is switched on.
URL	*<URL>*[3]	Optional. Editable text field. Empty by default. Supports: • Properties (page 392), e.g. '$[MY_URL]'. • System variables (page 392), e.g. '${MY_URL}'. • XPath expressions (page 392), e.g. $(upper-case('url')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Transport* is set to 'via WebService integration' and *Extended settings* is switched on.

1) Reference to master data record.
2) Secure password.
3) Valid URL of external web service.

Preconditions:

- SendFileService web service installed: if *Transport* is set to 'via WebService integration'.
- WebService Client adapter started: if *Transport* is set to 'via WebService integration'.
- Available process attachment.

Result:

- Current process attachment: unchanged.

Sets the following properties:

- SEEFX_REPORT := Error / Success message.
- SEEFX_STATUS := '200' in case of success, '204' in case of failure.
- SEEFX_TRANSFERID := The SEE FX transfer ID.
- STATUS := value of SEEFX_STATUS

Possible cases of failure:

Possible case of failure	Success or error behavior
WebService client adapter not started	Error. Throws acSEEFXFilesSend fault.
No connection to SEEFX server	Error. Throws acSEEFXFilesSend fault.
Invalid recipient (not found, not visible, or no access rights)	Error. Throws acSEEFXFilesSend fault.
Invlaid system variable in sender	Error. Throws acSEEFXFilesSend fault.

Policy: None.

1.20.4 acSEEFXInboxList

Caption: List SEE FX inbox.

Throws Fault=acSEEFXInboxList.

The *acSEEFXInboxList* activity is an SLMP based activity that provides a list of files that are contained in the BIS FileExchange inbox of the specified *User*. Uses the *listInbox* operation of the *SEEFX SLMP Client* adapter (refer to chapter *6.13 BIS Integration with SLMP (link)* sub-chapter *6.13.4 Available BIS FileExchange operations* of the *BIS* manual for more details).

Mode of operation:
Retrieves a list of all files in the BIS FileExchange inbox of the specified technical *User*. The response list of the SLMP client operation contains references to the files of the inbox. Depending on the selected *List method*, either a FILELIST object is created that contains all inbox file references and can be used as input for the *scForEach* (page 10) looping activity, or a new process is started for each inbox file reference. Please note that the "read" status of the listed inbox files is not changed.

Field name	Value	Description
SEEFX host	*<Ref>*[1]	Mandatory. Non-editable selection field. Master data BIS FileExchange Connection representing the SEEFX server.
User	*<Ref>*[1]	Mandatory. Editable selection field. Master data BIS FileExchange Personality Address or Bidirectional Address representing a technical SEEFX user. Supports: • Properties (page 392), e.g. '$[SEEFX_USER]'.

Field name	Value	Description
		• System variables (page 392), e.g. '${SEEFX_USER}'. • XPath expressions (page 392), e.g. $(upper-case('username')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000'))
List method		Mandatory. Non-editable selection field. Provides methods how to handle the SEE FX inbox file list.
	INVOKE	Invokes a new process for each list entry by creating MFT-SEEFXEvents (link).
	LIST	Creates a list file, i.e. creates and returns a FILELIST object (CSV file) containing all inbox files. This FILELIST object can be used as direct input to the activity providing a loop element.
Include read files	True	Retrieves a list of all files of the SEE FX inbox.
	False	Retrieves a list of unread files only.
Mapping parameter		Optional. Editable text field. Empty by default. Supports: • Properties (page 392), e.g. '$[LIST_MAPPING]'. • System variables (page 392), e.g. '${LIST_MAPPING}'. • XPath expressions (page 392), e.g. $(upper-case('mappingname')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000'))
Abort on error	True/False	Mandatory. Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Arbitrary text.

Preconditions:

- SLMP Client adapter started.

Result:

- New process attachment: FILELIST object (CSV file) containing all entries if *List method* is set to 'LIST'.
- MFT-Process for each inbox file if *List method* is set to 'INVOKE'.
- Current process attachment: unchanged if *List method* is set to 'INVOKE'.

Sets the following properties:

- LIST_COUNT := number of entries in the FILELIST object if *List method* is set to 'LIST'.
- FILE_COUNT := number of entries in the FILELIST object if *List method* is set to 'INVOKE'.

If the method is set to INVOKE and there are transferable files in the inbox, this activity spawns an MFT-Process for each file. The operation can return multiple transfers. A transfer can have multiple files, each file invoking its own MFT-Event process. The original payload before this activity was called will remain the same after this activity.

The spawned MFT-Process has the following properties set which can be used in MFT-Criteria:

- FROM_ADAPTER := 'SEE-FX’
- FROM_NAME := <creator of the file>
- FROM_SUBJECT := <subject of the transfer>
- FROM_PATH := <original file name>
- FROM_ID := <email address of file creator>

Possible cases of failure:

Possible case of failure	Success or error behavior
SEEFX server not accessible	Error. Throws acSEEFXInboxList fault.
Invalid property in *User*	Error. Throws acSEEFXInboxList fault.
Invalid system variable in *User*	Error. Throws acSEEFXInboxList fault.
Invalid property in *Mapping parameter*	Error. Throws acSEEFXInboxList fault.
Invalid system variable in *Mapping parameter*	Error. Throws acSEEFXInboxList fault.

Policies: none.

1.20.5 acSEEFXInboxPoll

Caption: Poll SEE FX inbox.

Throws Fault=acSEEFXInboxPoll.

The *acSEEFXInboxPoll* activity is an SLMP client based activity that polls the files that are contained in the BIS FileExchange inbox of the specified *User*. Uses the *pollInbox* operation of the *SEE FX SLMP Client* adapter (refer to chapter *6.13 BIS Integration with SLMP (link)* sub-chapter *6.13.4 Available BIS FileExchange operations* of the *BIS FileExchange Installation and Administration Guide* manual for more details).

Mode of operation:
Polls the inbox of the specified system user representing BIS assigned to the BIS SLMP endpoint. Each pollid file triggers a new incoming receiveTransfer event.

Filed name	Value	Description
SEEFX host	*<Ref>*[1]	Optional. Non-editable selection field. Master data BIS FileExchange Connection representing the SEEFX server.
User	*<Ref>*[1]	Mandatory. Editable selection field. Master data BIS FileExchange Personality Address or Bidirectional Address representing a technical SEE FX user whose inbox is being polled. Must be the name of a system user account assigned to SLMP endpoint that represents BIS. Supports: • Properties (page 392), e.g. '$[SEEFX_USER]'. • System variables (page 392), e.g. '${SEEFX_USER}'. • XPath expressions (page 392), e.g. $(upper-case('username')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000'))
Include read files	True	Retrieves all files of the SEEFX inbox.
	False	Default. Retrieves a list of unread files only i.e. all "new" files that are not marked as "read".

Filed name	Value	Description
		I All successfully polled files are automatically marked as "read".
Abort on error	True/False	Mandatory. Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Arbitrary text.

Preconditions:

- SLMP Client adapter started.

Results:

- Current process attachment: unused and unchanged.
- MFT-SEEFXTransferEvent: one event for ech polled file.
- Current process attachment: not used and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
No connection to SEEFX server	Error. Throws acSEEFXInboxPoll fault.
Invalid system variable in *User*	Error. Throws acSEEFXInboxPoll fault.

Policy: None.

1.20.6 acSEEFXMoveFile

Caption: SEEFXMoveFile.

Throws Fault=acSEEFXMoveFile.

The *acSEEFXMoveFile* activity is a web service based activity that moves a single file from a SEEFX source folder to a SEE FX target folder. Uses the *MoveFile* operation from the *FileTransferService3* SEE FX web service (refer to chapter *1.2 FileTransferService3* of the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to FileTransferService3Port web service.
Extended settings	True/False	Optional. Additional field group. Refer to Extended settings (page 304).

Field name	Value	Description
Source path	*<FilePath>*[2]	Mandatory. Editable text field. Default value '/My Files/'. Absolute path to the SEEFX source file to move. Must follow the SEEFX Paths guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_TARGET_PATH]'. • System variables (page 392), e.g. '${MY_TARGET_PATH}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
If file exists		Mandatory. Non-editable selection field. Provides supported strategies how to handle the case that the target file already exists.
	Auto rename	If a file with the file name specified in *Target path* already exists in the *Move to folder path*, a new file is created with a new name, usually generated from the original file name and a counter.
	Fail	If a file with the file name specified in *Target path* already exists in the *Move to folder path*, the activity fails. The process status is set to ERROR.
	Overwrite	If a file with the file name specified in *Target path* already exists in the *Move to folder path*, it is overwritten. The old file is moved to the recycle bin.
	Skip move	If a file with the file name specified in *Target path* already exists in the *Move to folder path*, the new file is not stored. The process state is set to SUCCESS. The process is continued with the next activity.
Move to folder path	*<FolderPath>*[3]	Optional. Editable text field. Default value '/My Files/SENT/'. Absolute path to the SEE FX target folder or file. Must follow the SEE FX Paths guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_TARGET_PATH]'. • System variables (page 392), e.g. '${MY_TARGET_PATH}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). • Moving files to another folder. • Renaming files in source folder.
Abort on error	True/False	Mandatory. Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Absolute SEEFX file path.
3) Absolute SEEFX folder or file path.
4) Arbitrary text.

Preconditions:

- FileTransferService3 web service installed.
- Web Services Client adapter started.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- SOURCE_PATH := The path of the source file.
- TARGET_PATH := The path of the target directory or file.
- TARGET_NAME := The name of the target file. Different from source file name in case of 'Auto rename' mode.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	**Mode**	**Success or error behavior**
WebService client adapter not started		Error. Throws acSEEFXMoveFile fault.
No connection to SEEFX server		Error. Throws acSEEFXMoveFile fault.
Wrong/Invalid URL	Enabled: *Extended settings*	Error. Throws acSEEFXMoveFile fault.
Wrong HTTP basic authentication	Enabled: *Extended settings*	Error. Throws acSEEFXMoveFile fault.
Non-existing system variable in *U RL*	Enabled: *Extended settings*	Error. Throws acSEEFXMoveFile fault.
Non-existing system variable in *User name*	Enabled: *Extended settings*	Error. Throws acSEEFXMoveFile fault.
Non-existing system variable in *Source path*		Error. Throws acSEEFXMoveFile fault.
Non-existing source file		Error. Throws acSEEFXMoveFile fault.
Locked source file		Error. Throws acSEEFXMoveFile fault.
Source file without permissions		Error. Throws acSEEFXMoveFile fault.
Non-existing target folder		Error. Throws acSEEFXMoveFile fault.
Non-existing system variable in *Move to folder path*		Error. Throws acSEEFXMoveFile fault.
Target file already exists	File exists mode: Auto rename	Success. File is renamed during move.
Target file already exists	File exists mode: Fail	Error. Throws acSEEFXMoveFile fault.
Target file already exists	File exists mode: Overwrite	Success. Target file is overwritten.
Target file already exists	File exists mode:Skip move	Success. Process is continued with next activity.

Policy: none.

1.20.7 acSEEFXNotificationsPoll

Caption: Poll SEE FX notifications.

Throws Fault=acSEEFXNotificationsPoll.

The *acSEEFXNotificationsPoll* activity is an SLMP based activity that polls files that have been sent to other SEE FX users. Uses the *pollNotifications* operation of the *FX SLMP Client* adapter (refer to chapter *6.13 BIS Integration with SLMP (link)* sub-chapter *6.13.4 Available BIS FileExchange operations* of the *BIS FileExchange Installation and Administration Guide* manual for more details).

Mode of operation:
Polls notifications from a process. No file is returned to the process. The polled notification will trigger a new incoming receiveNotifications event.

Field name	**Value**	**Description**
SEEFX host	*<Ref>*[1]	Optional. Non-editable selection field. Master data BIS FileExchange Connection representing the SEEFX server.
User	*<UserName>*[2]	Mandatory. Editable selection field. Empty by default. Master data BIS FileExchange Personality Address or Bidirectional Address representing a technical SEEFX user. Supports: • Properties (page 392), e.g. '$[SEEFX_USER]'. • System variables (page 392), e.g. '${SEEFX_USER}'. • XPath expressions (page 392), e.g. $(upper-case('username')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Abort on error	True/False	Mandatory. Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of existing SEEFX user.
3) Arbitrary text.

Preconditions:

- SLMP Client adapter started.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- MYKEY := acSEEFXNotificationsPoll.
- STATUS := '100'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
SEEFX server not accessible	Error. Throws acSEEFXNotificationsPoll fault.
Invalid property in *User*	Error. Throws acSEEFXNotificationsPoll fault.
Invalid system variable in *User*	Error. Throws acSEEFXNotificationsPoll fault.

Policy: none.

1.20.8 acSEEFXUploadFile2

Caption: SEEFX upload file (WebService).

Throws Fault=acSEEFXUploadFile2.

The *acSEEFXUploadFile2* activity is a web service based activity that uploads the current process attachment to the specified SEE FX account. Uses the *UploadFile2* operation from the *FileTransferService3* SEE FX web service (refer to chapter *1.2 FileTransferService3* of the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to FileTransferService3Port web service.
Extended settings	True/False	Optional. Additional field group. Default fields *Login with Token*, *URL* , *User name*, *Password* and *Login Token* are described in Extended settings with Token (page 305). Makes *Login with Token*, *URL*, *User name*, *Password* and *Login token* visible.
Target path	*<FilePath>*[2]	Optional. Editable text field. Default value '/My Files/'. Absolute path to the new SEEFX file. Must follow the SEEFX Path guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_TARGET_PATH]'. • System variables (page 392), e.g. '${MY_TARGET_PATH}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Description	*<Text>*[3]	Optional. Editable text field. Empty by default. Description for upload file. Supports: • Properties (page 392), e.g. '$[DESCRIPTION]'. • System variables (page 392), e.g. '${DESCRIPTION}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000'))
If file exists		Optional. Non-editable selection field. Non-editable selection field. Provides additional options for the file name of the target file and determines what to do in case there is already an existing file with the same name.

Field name	Value	Description
	Auto rename	Default. If a file with the file name specified in *Target path* already exists, a new file is created with a new name, usually generated from the original file name and a counter.
	Fail	If a file with the file name specified in *Target path* already exists, the activity fails. The process status is set to ERROR.
	Overwrite	If a file with the file name specified in *Target path* already exists, it is overwritten. The old file is moved to the recycle bin.
	Resume	If a file with the file name specified in *Target path* already exists, the new file is not stored. The process state is set to SUCCESS. The process is continued with the next activity.
Abort on error	True/False	Mandatory. Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Absolute SEEFX file path.
3) Arbitrary text.

Additional configuration:

System variable	Description
${MFT-MAX-RETRY}	Default value 0. If existing: max retry := <value>.
${MFT-WEBSERVICE-SO-TIMEOUT}	Default value 600. If existing: socket timeout := <value>. Keepalive := TRUE in each case.

Preconditions:

- FileTransferService3 web service installed.
- WebService client adapter started.
- Current process attachment available.

Result:

- Current process attachment: not changed.

Sets the following properties:

- TARGET_PATH := The path of the uploaded file.
- TARGET_NAME := The name of the uploaded file.

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
WebService client adapter not started		Error. Throws acSEEFXUploadFile2 fault.
No connection to SEEFX server		Error. Throws acSEEFXUploadFile2 fault.
Wrong/Invalid URL	Enabled *Extended settings*	Error. Throws acSEEFXUploadFile2 fault.
Wrong HTTP basic authentication	Enabled *Extended settings*	Error. Throws acSEEFXUploadFile2 fault.
Non-existing system variable in *U RL*	Enabled *Extended settings*	Error. Throws acSEEFXUploadFile2 fault.
Non-existing system variable in *User name*	Enabled *Extended settings*	Error. Throws acSEEFXUploadFile2 fault.
Path to directory instead of path to file		Error. Throws acSEEFXUploadFile2 fault.
Target file already existing	*File exists* mode: Fail	Error. Throws acSEEFXUploadFile2 fault.
Target file already existing	*File exists* modes: Auto rename, Overwrite	Success. File is uploaded. Process is continued with next activity.
Target file already existing	*File exists* mode: Resume	Success. File is not uploaded. Process is continued with next activity.
Target folder not existing		Success. Folder is created and file is uploaded.

Policy: None.

1.20.9 acSEEFXViewList2

Caption: SEEFX view file list.

Throws Fault=acSEEFXViewList2.

The *acSEEFXViewList2* activity is a web service based activity that lists file from a SEE FX account directory and returns a FILELIST object (CSV file) which can be used as input to the *scForEach* (page 10) activity. Uses the *ListContent3* operation from the *FileTransferService3* SEE FX web service (refer to chapter *1.2* of the manual for more details).

Mode of operation:
The *ListContent3* operation lists all specified files from the specified directory (and possibly sub-directories) and returns a list of file names and file paths. This list is converted into CSV format by the *BIC Client* adapter and can be used as input for the *scForEach* (page 10) looping activity.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. None-editable selection field. Provides master data external web services. Restricted to FileTransferService3Port web service.
Extended settings	True/False	Optional. Additional field group. Refer to Extended settings (page 304).

Field name	Value	Description
Source path	*<FolderPath>*[2]	Mandatory. Editable text field. Default value '/My Files/'. Absolute path to a SEE FX folder. Must follow the SEE FX Paths guidelines (page 398). The *Source path* supports: • Properties (page 392), e.g. '$[MY_SOURCE_PATH]'. • System variables (page 392), e.g. '${MY_SOURCE_PATH}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Include directories	True	Default. Include all files from the specified directory and from all sub-directories.
	False	Include only files from the specified directory.
Include pending files in list	True	Include pending files.
	False	Default. Exclude pending files.
Include virtual files in list	True	Include virtual files.
	False	Default. Exclude virtual files.
Mapping parameter		Optional. Editable text field. Empty by default. Mapping parameter for BIC conversion into CSV format. Available for the mapping as an AdditionalParameter with the name 'PARAMETER'. The *Mapping parameter* supports: • Properties (page 392), e.g. '$[MY_MAPPING_PARAM]'. • System variables (page 392), e.g. '${MY_MAPPING_PARAM}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000'))
Abort on error	True/False	Mandatory. Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Absolute SEEFX folder path.
3) Arbitrary text.

Preconditions:

- FileTransferService3 web service installed.
- Web Services Client adapter started.
- BIC Client adapter started.
- System variable ${MFT-FILELISTCREATOR} existing with name of mapping for conversion of file list into CSV format.
- Mapping from system variable above available in repository.
- Available process attachment.

Result:

- New process attachment: CSV file containing the file list.

Sets the following properties:

- LIST_COUNT := Number of entries in returned list.

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
WebService client adapter not started		Error. Throws acSEEFXViewList2 fault.
No connection to SEEFX server		Error. Throws acSEEFXViewList2 fault.
Wrong/Invalid URL	Enabled: *Extended settings*	Error. Throws acSEEFXViewList2 fault.
Wrong HTTP basic authentication	Enabled: *Extended settings*	Error. Throws acSEEFXViewList2 fault.
Non-existing system variable in *URL*	Enabled: *Extended settings*	Error. Throws acSEEFXViewList2 fault.
Non-existing system variable in *User name*	Enabled: *Extended settings*	Error. Throws acSEEFXViewList2 fault.
Invalid/not existing source path		Error. Throws acSEEFXViewList2 fault.
Source path is no directory		Error. Throws acSEEFXViewList2 fault.
Non-existing property in source path or mapping parameter		Error. Throws acSEEFXViewList2 fault.
Non-existing system variable in source path or mapping parameter		Error. Throws acSEEFXViewList2 fault.

Policy: None.

1.20.10 mdSEEFX

Caption: SEEFX upload.

Throws Fault=acForwardingSEEFX.

The *mdSEEFX* activity is a web service based single file activity that sends the current process attachment to the specified SEEFX target path. This actually represents a download. Uses the *UploadFile2* operation of the *FileTransferService2* SEEFX web service (refer to chapter *1.2 FileTransferService2* of the *BIS FileExchange WebService API HowTo* manual for more details).

The *acForwardingSEEFX* activity always uses the Consumed Web service with the name 'FileTransferService2Port'. The correct SEEFX server URL has to be specified in the 'Transport' tab.

The total number of recipients is restricted to only 1 (configured in the 'FileTransferService2Port' web service master data). An activity that provides an upload with more flexibility is *acSEEFXUploadFile2* (page 298).

Field name	Value	Description
Full path	*<FilePath>*[1]	Mandatory. Editable text field. Empty by default. Full target path for the upload file. Must follow the SEEFX Path guidelines (page 398), for instance '/My Files/test.txt'.

Field name	Value	Description
		Supports: • Properties (page 392), e.g. '$[MY_TARGET_PATH]' or '/My Files/$[TRACK_ID].dta'. • System variables (page 392), e.g. '${MY_TARGET_PATH}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Not all available SEEFX paths allow write operations. For instance, /My Subscriptions/ and /My Shares/ do not allow them.
On file exists		Mandatory. Non-editable selection field. Provides additional options for the file name of the target file and determines what to do in case there is already an existing file with the same name.
	Fail	Default. If a file with the file name specified in *Full path* already exists, the activity fails.
	Overwrite	If a file with the file name specified in *Full path* already exists, it is overwritten. The old file is moved to the recycle bin.
	Auto rename	If a file with the file name specified in *Full path* already exists, a new file is created with a new name.
Abort on error	True/False	Mandatory. Refer to Common Fields (page 3).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Absolute SEEFX file path.
2) Arbitrary text.

Preconditions:

- TransferFileService2 installed.
- WebService Client adapter started.
- Available process attachment.

Result:

- Current process attachment: not changed.

Sets the following properties:

- STATUS := 'SUCCESS_NOTIFY'.
- RESULT := 'Sent'.
- TO_ADAPTER := 'fileTransfer.uploadFile'.
- DIRECTION := 'OUT'.
- TO_NAME := The target file name.
- TARGET_NAME := The absolute target file path and name.

Possible cases of failure:

Possible case of failure	Success or error behavior
Missing access rights for web service user	Error. Throws *acForwardingSEEFX* fault.
Invalid target path	Error. Throws *acForwardingSEEFX* fault.
Invalid property in target path	Success. File is sent with '$[propname]' in its name.
Invalid system variable in target path	Error. Throws *acForwardingSEEFX* fault.
Already existing file name in case of *On file exists* set to 'Fail'	Error. Throws *acForwardingSEEFX* fault.

Policy:

REQUIREMENT/PAYLOAD-SCAN-DLP REQUIREMENT/PAYLOAD-ENCRYPT-TRANSPORT ALLOWANCE/TRANSPORT/SEEFX

1.20.11 Extended Settings

The Extended settings field group is used in all SEE FX related activities that are based on a SEE FX web service. These activities refer to this section.

Field name	Value	Description
Extended settings	True	Use extended settings. Displays additional fields *URL*, *Sender*, and *Password* which overwrite the settings from the *WebService endpoint*. Makes *URL*, *Sender*, and *Password* visible.
	False	Default. No additional fields displayed.
URL	*<URL>*[1]	Optional. Editable text field. Empty by default. Example: 'http://10.11.23.54:8080/portal-seefx/ws/FileTransferService3'. URL of the external web service. Overwrites the URL of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_SEEFX_URL]'. • System variables (page 392), e.g. '${MY_SEEFX_URL}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended settings* is switched on.
User name	*<Ref>*[2]	Optional. Editable text field. Empty by default. User name of HTTP basic authentication, represents a SEEFX user. Overwrites the HTTP authentication user name of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_SEEFX_USER]'. • System variables (page 392), e.g. '${MY_SEEFX_USER}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended settings* is switched on.

Field name	Value	Description
Password	*<Password>*[3]	Optional. Editable password field. Password of HTTP basic authentication. Overwrites the HTTP authentication password of the *WebService endpoint*. System variables and properties are not supported. Visible when *Extended settings* is switched on.

1) Valid URL to SEEFX web service.
2) Reference to master data.
3) Secure password.

1.20.12 Extended Settings with Token

The Extended settings field group is used in all SEEFX related activities that are based on a SEEFX web service. This is a special type providing an additional login mode using tokens which currently is supported by *acSEEFXDownloadFile* (page 283), *acSEEFXUploadFile2* (page 298), *acBISFXGetValueFor* (page 351), and *acBISFXSetValueFor* (page 353) activities.They refer to this section.

Field name	Value	Description
Extended settings	True	Use extended settings. Displays additional fields *Login with token*, *URL*, *Sender*, *Password*, and *Login token*. *URL*, *Sender*, and *Password* overwrite the settings from the *WebService endpoint*. Makes *Login with token*, *URL*, *Sender*, *Password*, and *Login token* visible.
	False	Default. No additional fields displayed.
Login with token		Switches between login modes. Visible when *Extended settings* is switched on.
	True	Use login token instead of user and password. Makes *Login token* visible.

Field name	Value	Description
	False	Use user and password for login. Makes *User name* and *Password* visible.
URL	*<URL>*[1]	Optional. Editable text field. Empty by default. Example: 'http://10.11.23.54:8080/portal-seefx/ws/FileTransferService3'. URL of the external web service. Overwrites the URL of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_SEEFX_URL]'. • System variables (page 392), e.g. '${MY_SEEFX_URL}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible when *Extended settings* is switched on.
User name	*<Ref>*[2]	Optional. Editable text field. Empty by default. User name of HTTP basic authentication, represents a SEEFX user. Overwrites the HTTP authentication user name of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_SEEFX_USER]'. • System variables (page 392), e.g. '${MY_SEEFX_USER}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible when *Extended settings* is switched on and *Login with token* is switched off.
Password	*<Password>*[3]	Optional. Editable password field. Password of HTTP basic authentication. Overwrites the HTTP authentication password of the *WebService endpoint*. System variables and properties are not supported. Visible when *Extended settings* is switched on and *Login with token* is switched off.
Login token	*<Token>*[4]	Optional. Editable text field. Empty by default. Valid session token from SEEFX server

1) Valid URL to SEEFX web service.
2) Reference to master data.
3) Secure password.
4) SEEFX server login token.

1.21 BISFX Activities

This section describes BISFX related activities which need a connection to a BISFX server. In contrast to SEEFX activities, only the WebService Client adapter is used. No SLMP connection between BISFX and BIS is needed.

Terminology and definitions

Key word	Abbr.	Description
Property		Structure that contains attributes for file and folder elements. Attributes are name, value, and ReadOnly flag.

#

1.21.1 Group Service

This section describes activities related to the *GroupService* BIS FX web service offering operations that create, modify and delete BIS FX groups.

The user that wants to access the *GroupService* needs the 'SEEIDM_WS' role, and the 'system' group must be the homegroup.

1.21.1.1 acBISFXAdministrationCreateGroup

Caption: BISFXAdministrationCreate Group.

Throws Fault=acBISFXAdministrationCreateGroup.

The *acBISFXAdministrationCreateGroup* activity is a web service based activity that creates a new group in BISFX. Uses the *CreateGroup* operation of the BISFX *GroupService* (refer to chapter *1.6 User and Groups Web Service (link)*, *section Operations*, subsection *Groups* of the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to GroupServicePort web service.
Group name	*<GroupName>*[2]	Mandatory. Editable text field. Empty by default. Name of the new BIS FX group. Supports: • Properties (page 392), e.g. '$[NEW_GROUP]'. • System variables (page 392), e.g. '${NEW_GROUP}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).

Field name	Value	Description
Owner group name	*<GroupName>*[3]	Mandatory. Editable text field. Empty by default. Name of an existing BIS FX group that owns the new BIS FX group. Supports: - Properties (page 392), e.g. '$[OWNER_GROUP]'. - System variables (page 392), e.g. '${OWNER_GROUP}'. - XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Allow new users	True	Creation of new users allowed in the new group.
	False	Default. Creation of new users not allowed in the new group.
Display name	*<Text>*[4]	Optional. Empty by default. Editable text field. The display name of the new BIS FX group. If this field is empty, the *Group name* is used as display name, too. Supports: - Properties (page 392), e.g. '$[NEW_GROUP_NAME]'. - System variables (page 392), e.g. '${NEW_GROUP_NAME}'. - XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Template user name	*<UserName>*[5]	Optional. Empty by default. Editable text field. The name of the user which is to be used as template when creating the other users. Supports: - Properties (page 392), e.g. '$[TEMPLATE_USER]'. - System variables (page 392), e.g. '${TEMPLATE_USER}'. - XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Home group name	*<GroupName>*[3]	Optional. Empty by default. Editable text field. The name of the BISFX home group. Supports: - Properties (page 392), e.g. '$[HOME_GROUP]'. - System variables (page 392), e.g. '${HOME_GROUP}'. - XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Allowed users	*<Integer>*[6]	Mandatory. Editable number field. Default value '10'. Maximum value '9999'. The number of users that are allowed in the new group. Properties and system variables are not supported.
Extended settings	True/False	Optional. Additional field group with *URL*, *Sender*, and *Password* fields. Refer to *Extended settings* (page 360), standard type.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of a new BISFX group.
3) Name of an existing BISFX group.
4) Arbitrary text.
5) Name of an existing BIS user.
6) Positive Integer number.

Preconditions:

- BISFX GroupService web service installed.
- WebService client adapter started.

Result:

- Current process attachment: unused and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationCreateGroup fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationCreateGroup fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationCreateGroup fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationCreateGroup fault.
Unsufficient rights for user	Error. Throws acBISFXAdministrationCreateGroup fault.
Non-existing group in *Owner group name*	Error. Throws acBISFXAdministrationCreateGroup fault.
Non-existing group in *Home group name*	Error. Throws acBISFXAdministrationCreateGroup fault.
Non-existing template user in *Template user name*	Error. Throws acBISFXAdministrationCreateGroup fault.
Already existing group with name from *Group name*	Error. Throws acBISFXAdministrationCreateGroup fault.

Policies: none.

1.21.1.2 acBISFXAdministrationChangeGroup

Caption: BISFXAdministrationGetGroup.

Throws Fault=acBISFXAdministrationChangeGroup.

The *acBISFXAdministrationGetGroup* activity is an activity that changes the properties of an existing BIS FX group. Uses the *ChangeGroup* operation of the BIS FX *GroupService* (refer to chapter *1.6 User and Groups Web Service (link)*, *section Operations* subsection *Groups* of the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to GroupServicePort web service.
Group name	*<GroupName>*[2]	Mandatory. Editable text field. Empty by default. Name of the group whose properties are to be changed. Supports: • Properties (page 392), e.g. '$[MY_GROUP]'. • System variables (page 392), e.g. '${MY_GROUP}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Owner Group name	*<GroupName>*[2]	Optional. Editable text field. Empty by default. Name of the owner group. Supports: • Properties (page 392), e.g. '$[OWNER_GROUP]'. • System variables (page 392), e.g. '${OWNER_GROUP}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Allow new users	True	New users allowed.
	False	Default. New users not allowed.
Display name		Optional. Editable text field. Empty by default. Display name for the group.
Template user name	*<UserName>*[3]	Optional. Editable text field. Empty by default. The name of existing user which is to be used as template for other users of this group. Supports: • Properties (page 392), e.g. '$[TEMPLATE_USER]'. • System variables (page 392), e.g. '${TEMPLATE_USER}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Home group name	*<GroupName>*[2]	Optional. Editable text field. Empty by default. The name of the home group. Supports: • Properties (page 392), e.g. '$[HOME_GROUP]'. • System variables (page 392), e.g. '${HOME_GROUP}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Allowed users	*<Integer>*[4]	Optional. Editable number field. Default value '10'. Maximum value '9999'. The number of users that are allowed in the group.

Field name	Value	Description
		Properties and system variables are not supported.
Extended settings	True	Optional. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *Sender*, and *Password* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[5]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of existing SEEFX group.
3) Name of existing SEEFX user.
4) Positive Integer number.
5) Arbitrary text.

Preconditions:

- GroupService web service installed.
- WebService client adapter started.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
System variable not existing	Error. Throws acBISFXAdministrationChangeGroup fault.

Policies: none.

1.21.1.3 acBISFXAdministrationDeleteGroup

Caption: BISFXAdministrationDelete Group.

Throws Fault=acBISFXAdministrationDeleteGroup.

The *acBISFXAdministrationDeleteGroup* activity is a web service based activity that deletes an existing group in BISFX. Uses the *DeleteGroup* operation of the BISFX *GroupService* (refer to chapter *1.6 User and Groups Web Service (link)*, *section Operations*, subsection *Groups* of the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*^{1}	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to GroupServicePort web service.
Group name	*<GroupName>*^{2}	Mandatory. Editable text field. Empty by default. Name of the BISFX group to be deleted. Supports: • Properties (page 392), e.g. '$[NEW_GROUP]'. • System variables (page 392), e.g. '${NEW_GROUP}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')).
Extended settings	True/False	Optional. Additional field group with *URL*, *UserName*, and *Password* fields. Refer to *Extended settings* (page 360), standard type.
Queue	*<Ref>*^{1}	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*^{3}	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of an existing BISFX group.
3) Arbitrary text.

Preconditions:

- BISFX GroupService web service installed.
- WebService client adapter started.

Result:

- Current process attachment: unused and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationCreateGroup fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationCreateGroup fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationCreateGroup fault.
Non-existing group in *Group name*	Error. Throws acBISFXAdministrationCreateGroup fault.
System variable in *Group name* not existing	Error. Throws acBISFXAdministrationCreateGroup fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationCreateGroup fault.
Unsufficient BISFX rights for user	Error. Throws acBISFXAdministrationCreateGroup fault.

Policies: none.

1.21.1.4 acBISFXAdministrationGetGroup

Caption: BISFXAdministartionGetGroup.

Throws Fault=acBISFXAdministrationGetGroup.

The *acBISFXAdministrationGetGroup* activity is a web service based activity that retrieves information about a BIS FX group. Uses the *GetGroup* operation of the BIS FX *GroupService* (refer to chapter *1.6 User and Groups Web Service (link)*, *section Operations,* subsection *Groups* of the *BIS FileExchange Web Service API* manual manual for more details).

Mode of operation:
Retrieves information about the specified BIS FX group and stores it as current process attachment in XML format.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to GroupServicePort web service.
Group name	*<GroupName>*[2]	Mandatory. Empty by default. Editable text field. Name of the BIS FX group. Supports: • Properties (page 392), e.g. '$[BISFX_GROUP]'. • System variables (page 392), e.g. '${BISFX_GROUP}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Extended settings	True/False	Optional. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *Sender*, and *Password* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).

Field name	Value	Description
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of existing SEEFX group.
3) Arbitrary text.

Preconditions:

- GroupService web service installed.
- WebService client adapter started.

Result:

- New process attachment: XML format, UTF-8, contains BIS FX group information (refer to example file content below).

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong web service configuration	Error. Throws acBISFXAdministrationGetGroup fault.
Unsufficient rights	Error. Throws acBISFXAdministrationGetGroup fault.
Non-existing property in user name	Error. Tries to get group with non-resolved name. Throws acBISFXAdministrationGetGroup fault.
Non-existing system variable in user name	Error. Throws acBISFXAdministrationGetGroup fault.
Non-existing group name	Error. Throws acBISFXAdministrationGetGroup fault.

Policies: none.

Example of Group Information Attachment

```
<ns2:GetGroupResponse xmlns:ns2="http://uri.seeburger.com/seefx/groupService">
    <Group>
        <GroupName>InternalUsers</GroupName>
        <OwnerGroupName>system</OwnerGroupName>
        <AllowNewUsers>true</AllowNewUsers>
        <DisplayName>InternalUsers</DisplayName>
    </Group>
</ns2:GetGroupResponse>
```

1.21.1.5 acBISFXAdministrationAddRoles

Caption: BISFXAdministrationAddRoles.

Throws Fault=acBISFXAdministrationAddRoles.

The *acBISFXAdministrationAddRoles* activity is a web service based activity that adds a list of roles to a group. Uses the *AddRoles* operation of the SEE FX GroupService web service (refer to chapter *1.6 User and Groups*

Web Service (link), *section Operations*, subsection *Groups* of the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to GroupServicePort web service.
Groupname	*<GroupName >*[2]	Mandatory. Editable text field. Empty by default. Name of an existing group. Supports: • Properties (page 392), e.g. '$[GROUP_NAME]'. • System variables (page 392), e.g. '${GROUP_NAME}'.
New group roles	*<RolesList>*[3]	Mandatory. Editable text field. Empty by default. Semicolon-separated list of BISFX group application roles (refer to *Application roles*(link) in the BISFX Identity Management manual for more details). Supports: • Properties (page 392), e.g. '$[GROUP_ROLES]' or '$[GROUP_ROLE1];$[GROUP_ROLE2];$[GROUP_ROLE3]'. • System variables (page 392), e.g. '${GROUP_ROLES}'.
Extended settings	True/False	Optional. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *Sender*, and *Password* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of existing SEEFX group.
3) Semicolon separated list of SEEFX role names.
4) Arbitrary text.

Preconditions:

- GroupService web service installed.
- WebService client adapter started.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationAddRoles fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationAddRoles fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationAddRoles fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationAddRoles fault.
Unsufficient rights for user	Error. Throws acBISFXAdministrationAddRoles fault.
System variable not existing	Error. Throws acBISFXAdministrationAddRoles fault.
Unknown group	Error. Throws acBISFXAdministrationAddRoles fault.
Unknown group application role	Error. Throws acBISFXAdministrationAddRoles fault.
Valid but unassigned role	Success. No error on BISFX server.

Policies: none.

1.21.1.6 acBISFXAdministrationRemoveRoles

Caption: BISFXAdministrationRemoveRoles.

Throws Fault=acBISFXAdministrationRemoveRoles.

The *acBISFXAdministrationRemoveRoles* activity is a web service based activity that removes roles from a group. Uses the *RemoveRoles* operation of the SEE FX GroupService web service (refer to chapter *1.6 User and Groups Web Service (link)*, *section Operations*, subsection *Groups* of the *BIS* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to GroupServicePort web service.
Group name	*<GroupName>*[2]	Mandatory. Editable text field. Empty by default. Name of an existing group. Supports: • Properties (page 392), e.g. '$[GROUP_NAME]'. • System variables (page 392), e.g. '${GROUP_NAME}'.
New group roles	*<List>*[3]	Mandatory. Editable text field. Empty by default. Semicolon-separated list of roles. Supports: • Properties (page 392), e.g. '$[GROUP_ROLES]' or '$[GROUP_ROLE1];$[GROUP_ROLE2];$[GROUP_ROLE3]'. • System variables (page 392), e.g. '${GROUP_ROLES}'.
Extended settings	True/False	Optional. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *Sender,* and *Password* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields.
Abort on error	True/False	Refer to Common Fields.
Log message	*<Text>*[4]	Optional. Refer to Common Fields.

1) Reference to master data record.
2) Name of existing SEEFX group.
3) Semicolon separated list of SEEFX roles.
4) Arbitrary text.

Preconditions:

- GroupService web service installed.
- WebService client adapter started.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
System variable not existing	Error. Throws acBISFXAdministrationRemoveRoles fault.

Policies: none.

1.21.1.7 acBISFXAdministrationAddGroupMembers

Caption: BISFXAdministrationAddGroupMembers.

Throws Fault=acBISFXAdministrationAddGroupMembers.

The *acBISFXAdministrationAddGroupMembers* activity is a web service based activity that adds one or more users to a group. Uses the *AddGroupMembers* operation of the BISFX *GroupService* web service (refer to chapter *1.6 User and Groups Web Service (link)*, *section Operations*, subsection *Groups* of the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to GroupServicePort web service.
Group name	*<GroupName>*[2]	Mandatory. Editable text field. Empty by default. Name of an existing group. Supports: • Properties (page 392), e.g. '$[GROUP_NAME]'. • System variables (page 392), e.g. '${GROUP_NAME}'.
New group members	*<List>*[3]	Mandatory. Editable text field. Empty by default. Semicolon-separated list of users. Supports: • Properties (page 392), e.g. '$[GROUP_MEMBERS]' or '$[GROUP_MEMBER1];$[GROUP_MEMBER2];$[GROUP_MEMBER3]'. • System variables (page 392), e.g. '${GROUP_MEMBERS}'.
Extended settings	True/False	Optional. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *Sender*, and *Password* visible.

Field name	Value	Description
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of existing BISFX group.
3) Semicolon separated list of BISFX user names.
4) Arbitrary text.

Preconditions:

- GroupService web service installed.
- WebService client adapter started

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationAddGroupMembers fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationAddGroupMembers fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationAddGroupMembers fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationAddGroupMembers fault.
Unsufficient rights for user	Error. Throws acBISFXAdministrationAddGroupMembers fault.
System variable not existing	Error. Throws acBISFXAdministrationAddGroupMembers fault.

Policies: none.

1.21.1.8 acBISFXAdministrationAddVisibleGroups

Caption: BISFXAdministrationAddVisibleGroups.

Throws Fault=acBISFXAdministrationAddVisibleGroups.

The *acBISFXAdministrationAddVisibleGroups* activity is a web service based activity that adds one or more groups to the list of groups visible to the members of a group. Uses the *AddVisibleGroups* operation of the SEE FX GroupService web service (refer to chapter *1.6 User and Groups Web Service (link)*, *section Operati ons*, subsection *Groups* of the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to GroupServicePort web service.
Group name	*<GroupName>*[2]	Mandatory. Editable text field. Empty by default. Name of an existing group. Supports: • Properties (page 392), e.g. '$[GROUP_NAME]'. • System variables (page 392), e.g. '${GROUP_NAME}'.
Groups to see	*<GroupsList>*[3]	Mandatory. Editable text field. Empty by default. Semicolon separated list of group names. Groups to be marked as visible for the members of the *Group* above. Supports: • Properties (page 392), e.g. '$[GROUPS]' or '$[GROUP1];$[GROUP2];$[GROUP3]'. • System variables (page 392), e.g. '${GROUS}'. Whether only the members of these groups are visible to the *Group* members or whether these groups are visible, too, is controlled by the *Is visible* flag below.
Is visible	True	Group and group members are visible.
	False	Default. Only group members are visible.
Extended settings	True/False	Optional. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *Sender*, and *Password* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of an existing BISFX group.
3) Semicolon separated list of BISFX group names.
4) Arbitrary text.

Preconditions:

- GroupService web service installed.
- WebService client adapter started.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationAddVisibleGroups fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationAddVisibleGroups fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationAddVisibleGroups fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationAddVisibleGroups fault.
Unsufficient rights for user	Error. Throws acBISFXAdministrationAddVisibleGroups fault.
System variable not existing	Error. Throws acBISFXAdministrationAddVisibleGroups fault.
Group not found	Success. BISFX sends no response for the *AddVisibleGroups* operation, neither an error response, although the BISFX server logs an error.
Group added twice	Success. No error on the BISFX server.

Policies: none.

1.21.2 User Service

This section describes activities related to the *UserService* BIS FX web service offering operations that create, modify and delete BIS FX users.

The user that wants to access the *UserService* needs the 'SEEIDM_WS' role, and the 'system' group must be the homegroup.

1.21.2.1 acBISFXAdministrationCreateUser

Caption: BISFXAdministrationCreate User.

Throws Fault=acBISFXAdministrationCreateUser.

The *acBISFXAdministrationCreateUser* activity is a web service based activity that creates a new BISFX user. Uses the *CreateUser* operation from the *UserService* BISFX web service (refer to chapter *1.6 User and Groups Web Service (link)* in the *BIS FileExchange WebService API HowTo* manual for more details).

Mode of operation:
Creates a new BISFX user account. Implicitly, the new account has the following settings: accountExpired=false, accountLocked=false, enabled=true, externalAuthentication=false.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to UserServicePort web service.
User name	*<UserName>*[2]	Mandatory. Editable selection field. Master data BISFX address, own or partner or both. Name of existing BISFX user. Supports: • System variables, e.g. '${BISFX_USERNAME}'. • Properties, e.g. '$[BISFX_USERNAME]'.
Email address	*<EMailAddress>*[3]	Mandatory. Editable text field. Empty by default. Email address for the BISFX user. Supports: • System variables, e.g. '${EMAIL}'. • Properties, e.g. '$[EMAIL]'.
Display name	*<Text>*[4]	Optional. Editable text field. Empty by default. Name shown for the new BISFX user Supports: • System variables, e.g. '${BISFX_USERNAME}'. • Properties, e.g. '$[BISFX_USERNAME]'.
User password from property		Password handling configuration. Enables switching between standard password field and text field with property and system variable support. Visible when *External authentication* is switched off.
	True	Use a text field for password input enabling the use of properties.

Field name	Value	Description
	False	Default. Use standard password field.
User password	*<Password>*[5]	Optional. Password BISFX user authentication. • Editable password field if *User password from property* is switched off. Properties and system variables are not supported. • Editable text field if *User password from property* is switched on. Properties and system variables are supported. Visible when *External authentication* is switched off.
Settings from template		Provides account data from either template or manual input.
	True	Default. Uses account data from user template. Makes *Template user name* visible.

Field name	Value	Description
	False	Enter account data directly without user template. Makes the following fields visible: *Groups to add user to*, *Home group name*, *Credentials expired*, *Email notification*, *Account expires*.
Template user name	*<UserName>*[6]	Mandatory. Editable text field. Empty by default. Name of a user used as template for this user. Supports: - System variables, e.g. '${USER_NAME}'. - Properties, e.g. '$[USER_NAME]'. Visible when *Template user* is switched on.
Groups to add user to	*<GroupsList>*[7]	Optional. Editable selection field. Semicolon separated list of group names. Supports: - System variables, e.g. '${BISFX_GROUP}'. - Properties, e.g. '$[BISFX_GROUP]'.
Home group name	*<GroupName>*[8]	Optional. Editable selection field. Master data BISFX addresses. Name of home group. Supports: - System variables, e.g. '${BISFX_GROUP}'. - Properties, e.g. '$[BISFX_GROUP]'.
Credentials expired	True	The password is expired. A new password must be entered at the next login.
	False	Default. The password is valid.
Email notification	True	User will get email notifications.
	False	Default. No email notifications sent.
Account expires	True	User account has an expiration date. Makes *Expires in days* visible.

Field name	**Value**	**Description**
	False	Default. No expiration of user account.
Account expires in days	*<Integer>*[9]	Mandatory. Editable number field. Default value '30'. Number of days until the user account expires. Properties and system variables are not supported. Visible when *Account expires* is switched on.
Extended settings	True/False	Optional. Extended settings for *WebService endpoint*. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *User name*, and *Password* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Valid new BISFX user name.
3) Valid email address.
4) Arbitrary text.
5) Secure password.
6) Name of an existing BISFX template user.
7) Semicolon separated list of BISFX group names.
8) Name of existing BISFX group.
9) Positive Integer number.

Preconditions:

- WebService Client adapter started.
- UserService web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties: None.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationCreateUser fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationCreateUser fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationCreateUser fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationCreateUser fault.
Unsufficient rights for user	Error. Throws acBISFXAdministrationCreateUser fault.
System variable not existing	Error. Throws acBISFXAdministrationCreateUser fault.

Policies: none.

1.21.2.2 acBISFXAdministrationChangeUser

Caption: BISFXAdministrationChange User.

Throws Fault=acBISFXAdministrationChangeUser.

The *acBISFXAdministrationChangeUser* activity is a web service based activity that changes email address, display name and password of an existing BISFX user. Uses the *ChangeUser* operation from the *UserService* BISFX web service (refer to chapter *1.6 User and Groups Web Service (link)* in the *BIS FileExchange WebService API HowTo* manual for more details).

Mode of operation:
The specified user must exist. The *acBISFXAdministrationChangeUser* activity can change email address, display name and/or password: If the appropriate field is non-empty, this value is entered to the BISFX account. If it is empty, nothing is done. Values cannot be deleted by *acBISFXAdministrationChangeUser* but only modified.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to UserServicePort web service.
User name	*<UserName>*[2]	Mandatory. Editable selection field. Master data BISFX address, own or partner or both. Name of existing BISFX user. Supports: • System variables, e.g. '${BISFX_USERNAME}'. • Properties, e.g. '$[BISFX_USERNAME]'.
Email address	*<EMailAddress>*[3]	Optional. Editable text field. Empty by default. Email address for the BISFX user. An empty value is ignored. A non-empty value will replace the current email address. Supports: • System variables, e.g. '${EMAIL}'. • Properties, e.g. '$[EMAIL]'.
Display name	*<Text>*[4]	Optional. Editable text field. Empty by default. Name shown for the new BISFX user. An empty value is ignored. A non-empty value will replace the current display name. Supports: • System variables, e.g. '${BISFX_USERNAME}'.

Field name	Value	Description
		• Properties, e.g. '$[BISFX_USERNAME]'.
User password from property		Password handling configuration. Enables switching between standard password field and text field with property and system variable support. Visible when *External authentication* is switched off.
	True	Use a text field for password input enabling the use of properties.
	False	Default. Use standard password field.
User password	*<Password>*[5]	Optional. Password BISFX user authentication. An empty value is ignored. A non-empty value will replace the current password. • Editable password field if *User password from property* is switched off. Properties and system variables are not supported. • Editable text field if *User password from property* is switched on. Properties and system variables are supported. Visible when *External authentication* is switched off.
Email notification		Mandatory. Non-editable selection field for notification configuration changes.
	Keep	Default. Do not change the current configuration.
	No	Disable email notification.
	Yes	Enable email notification.
Extended settings	True/False	Optional. Extended settings for *WebService endpoint*. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *User name*, and *Password* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Valid BISFX user name.
3) Valid email address.
4) Arbitrary text.
5) Secure password.

Preconditions:

- WebService Client adapter started.
- UserService web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties: None.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationChangeUser fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationChangeUser fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationChangeUser fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationChangeUser fault.
Unsufficient rights for user	Error. Throws acBISFXAdministrationChangeUser fault.
System variable not existing	Error. Throws acBISFXAdministrationChangeUser fault.

Policies: none.

1.21.2.3 abBISFXAdministrationDeleteUser

Caption: BISFXAdministrationDelete User.

Throws Fault=acBISFXAdministrationDeleteUser.

The *acBISFXAdministrationDeleteUser* activity is a web service based activity that deletes an existing BISFX user. Uses the *DeleteUser* operation from the *UserService* BISFX web service (refer to chapter *1.6 User and Groups Web Service (link)* in the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to UserServicePort web service.
User name	*<UserName>*[2]	Mandatory. Editable selection field. Master data BISFX addresses. Name of existing BIXFX user. Supports: • System variables, e.g. '${BISFX_USER}'. • Properties, e.g. '$[BISFX_USER]'.
Extended settings	True/False	Optional. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *User name*, and *Pass word* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of an existing BISFX user.
3) Arbitrary text.

Preconditions:

- WebService Client adapter started.
- UserService web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties: None.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationCreateUser fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationCreateUser fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationCreateUser fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationCreateUser fault.
Unsufficient rights for user	Error. Throws acBISFXAdministrationCreateUser fault.
System variable not existing	Error. Throws acBISFXAdministrationCreateUser fault.

Policies: none.

1.21.2.4 acBISFXAdministrationGetUser

Caption: BISFXAdministrationGetUser.

Throws Fault=acBISFXAdministrationGetUser.

The *acBISFXAdministrationGetUser* activity is a web service based activity that retrieves information about a BISFX user. Uses the *GetUser* operation of the *UserService* BISFX web service (refer to chapter *1.6 User and Groups Web Service (link)* of the *BIS FileExchange Web Service API* manual for more details).

Mode of operation:
Retrieves information about the specified BISFX user and stores it as current process attachment in XML format.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to UserServicePort web service.
User name	*<UserName>*[2]	Mandatory. Editable text field. Name of the BISFX user. Supports:

Field name	Value	Description
		• Properties (page 392), e.g. '$[BISFX_USER]'. • System variables (page 392), e.g. '${BISFX_USER}'.
Extended settings	True/False	Optional. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *User name*, and *Password* visible.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of existing BISFX user.
3) Arbitrary text.

Preconditions:

- WebService Client adapter started.
- UserService web service installed.

Result:

- New process attachment: XML format, UTF-8, contains BIS FX user information (refer to example file content below).

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong web service configuration	Error. Throws acBISFXAdministrationGetUser fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationGetUser fault.
no connection toBISFX server	Error. Throws acBISFXAdministrationGetUser fault.
Invalid URL in *Extended settings*	Error. Throws acBISFXAdministrationGetUser fault.
Unsufficient rights	Error. Throws acBISFXAdministrationGetUser fault.
Non-existing property in *User name*	Error. Tries to get user with non-resolved name. Throws acBISFXAdministrationGetUser fault.
Non-existing system variable in *User name*	Error. Throws acBISFXAdministrationGetUser fault.
Non-existing user name in *User name*	Error. Throws acBISFXAdministrationGetUser fault.

Policies: none.

Example of User Information Attachment

```
<ns2:GetUserResponse xmlns:ns2="http://uri.seeburger.com/seefx/userService">
   <return>
        <AccountExpired>false</AccountExpired>
        <AccountLocked>false</AccountLocked>
        <CredentialsExpired>false</CredentialsExpired>
        <DisplayName>Operator1</DisplayName>
        <EmailAddress>operator1@maildomain.com</EmailAddress>
        <EmailNotification>true</EmailNotification>
        <Enabled>true</Enabled>
        <ExpirationDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:nil="true"/>
        <ExternalAuthentication>false</ExternalAuthentication>
        <HomeGroupName>ExternalUsers</HomeGroupName>
        <InvitationExpirationDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:nil="true"/>
        <Language>en</Language>
        <TemplateUser>true</TemplateUser>
        <Timezone>Europe/Berlin</Timezone>
        <UserName>operator1</UserName>
    </return>
</ns2:GetUserResponse>
```

1.21.2.5 acBISFXAdministrationAddUserToGroups

Caption: BISFXAdministrationAddUserToGroups.

Throws Fault=acBISFXAdministrationAddUserToGroups.

The *acBISFXAdministrationAddUserToGroups* activity is a web service based activity that adds a BISFX user to one or several BISFX groups. Uses the *AddUserToGroups* operation from the *UserService* BISFX web service (refer to chapter *1.6 User and Groups Web Service (link)* from the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to UserServicePort web service.
User name	*<UserName>*[2]	Mandatory. Editable selection field. Empty by default. The BISFX user name. Supports: • System variables, e.g. '${BISFX_USERNAME}'. • Properties, e.g. '$[BISFX_USERNAME]'.
Groups to add user to	*<GroupsList>*[3]	Mandatory. Editable selection field. Semicolon separated list of BISFX groupnames. Supports: • System variables, e.g. '${BISFX_GROUP}'. • Properties, e.g. '$[BISFX_GROUP]'.
Extended settings	True/False	Optional. Additional field group. Refer to *Extended settings* (page 360), properties supporting standard type. Makes *URL*, *User name*, and *Password* visible.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of an existing BISFX user.
3) Semicolon separated list of BISFX group names.
4) Arbitrary text.

Preconditions:

- WebService Client adapter started.
- UserService web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationAddUser fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationAddUser fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationAddUser fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationAddUser fault.
Unsufficient rights for user	Error. Throws acBISFXAdministrationAddUser fault.
System variable not existing	Error. Throws acBISFXAdministrationAddUser fault.

Policies: none.

1.21.2.6 acBISFXAdministrationAddUserSSHKey

Caption: BISFXAdministrationAdd UserSSHKey.

Throws Fault=acBISFXAdministrationAddUserSSHKey.

The *acBISFXAdministrationAddUser*SSHKey activity is a web service based activity that adds an SSH key to an existing BISFX user. Uses the *AddUserSSHKey* operation from the *UserService* BISFX web service (refer to chapter *1.6 User and Groups Web Service (link)* in the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to UserServicePort web service.
User name	*<UserName>*[2]	Mandatory. Editable selection field. Empty by default. Master data BISFX own address or partner address. Name of existing BIS FX user. Supports: • System variables, e.g. '${BISFX_USER}'. • Properties, e.g. '$[BISFX_USER]'.
Decrypt alias	*<Ref>*[1]	Optional. Non-editable text field with Keystore Manager selection dialog. Supports RSA and DSA signature algorithms only. Other algorithms are not supported.
Extended settings	True/False	Optional. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *User name*, and *Pass word* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Name of existing BISFX user.
3) Arbitrary text.

Preconditions:

- WebService Client adapter started.
- UserService web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties: None.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationCreateUser fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationCreateUser fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationCreateUser fault.
Unsupported signature algorithm	Error. Throws acBISFXAdministrationCreateUser fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationCreateUser fault.
Unsufficient rights for user	Error. Throws acBISFXAdministrationCreateUser fault.
System variable in *User name* not existing	Error. Throws acBISFXAdministrationCreateUser fault.

Policies: none.

1.21.3 Share Management Service

This section describes activities related to the *ShareManagementService3* BIS FX web service.

1.21.3.1 acBISFXShareManagmentShareFolder

Caption: BISFXShareManagmentShareFolder.

Throws Fault=acBISFXShareManagmentShareFolder.

The *acBISFXShareManagmentShareFolder* activity is a web service based activity that shares a folder of a SEEFX account with a single recipient. Uses the *S*hareFolder operation from the *ShareManagementService3* BIS FX web service (refer to chapter *1.4 ShareManagementService3 (link)*, sub chapter *1.4.2 Operations* in the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to ShareManagementService3Port web service.
Full folder path	*<FolderPath>*[2]	Mandatory. Editable text field. Default value '/My Files/'. The absolute path of the folder to be shared. Must follow the SEE FX Path Guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_SHARE_FOLDER]'. • System variables (page 392), e.g. '${MY_SHARE_FOLDER}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
User type		Mandatory. Editable selection field. Default value 'User'. The type of the share recipient. Supports: • Properties (page 392), e.g. '$[REC_TYPE]'.

Field name	Value	Description
		- System variables (page 392), e.g. '${REC_TYPE}'.
	Group	SEE FX group.
	User	Default. SEE FX user.
User name	*<Text>*[3]	Mandatory. Editable selection box. Name of the share recipient. Supports: - Properties (page 392), e.g. '$[MY_RECIPIENT]'. - System variables (page 392), e.g. '${MY_RECIPIENT}'. - XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
User rights		Mandatory. Editable selection field. Default value 'View'. The access rights for the share recipient for this share folder.
	Delete	Allows to: - delete files or folders.
	Permit	Allows to: - create permits for this folder. - create download permits if combined with 'Read'. - create upload permits if combined with 'Write'.
	Read	Allows to: - see the content of the folder. - copy files. - download files.
	Read;View	Allows to: - see content of the folder. - copy files. - download files. - view documents (saving, printing, or forwarding not allowed).
	Read;Write;Delete;View	Includes all rights except permit related rights.
	View	Default. Allows to: - view only documents: The document can only be viewed in the brower, it is not possible to save, to print, or to forward the document.
	Write	Allows to: - copy or move own files to this folder. - edit existing files. - upload files. - create folders. - edit folders.
Display name	*<Text>*[3]	Optional. Editable text field. Empty by default. Display name of the share recipient. Supports: - Properties (page 392), e.g. '$[MY_RECIPIENT_NAME]'. - System variables (page 392), e.g. '${MY_RECIPIENT_NAME}'.

Field name	Value	Description
		• XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Home group	<GroupName>	Optional. Editable text field. Empty by default. Home group of the share recipient in case of 'User' type. Supports: • Properties (page 392), e.g. '$[REC_HOME_GROUP]'. • System variables (page 392), e.g. '${REC_HOME_GROUP}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Is subscribed	True	Subscription of this folder by the share recipient.
	False	Default. No subscription of this folder by the share recipient.
EMail address	*<EMailAddress>*[4]	Optional. Editable text field. Empty by default. The EMail address of the share recipient in case of 'User' type. Supports: • Properties (page 392), e.g. '$[EMAIL]'. • System variables (page 392), e.g. '${EMAIL}'.
Apply to subfolders	True	Apply the access rights also to the subfolders.
	False	Default. Apply the access rights to the specified folder only.
Send notification	True	Set the notification flag.
	False	Default. Do not set the notification flag.
Automatic subscription	True	Set auto subscription mode. Be sure to have sufficient rights for auto-subsription settings otherwise this activity fails.
	False	Default. Do not set auto subscription mode.
Extended settings	True/False	Optional. Use extended settings for *WebService endpoint*. Additional field group. Refer to *Extended settings* (page 360), properties supporting type. Makes *URL*, *User name*, *Password from property*, and *Password* visible.
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Absolute SEEFX folder path.
3) Arbitrary text.
4) Valid syntactically correct EMail address.
5) Valid URL to SEE FX web service.
6) Secure password.

Preconditions:

- WebService Client adapter started.
- ShareManagementService3 web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- TARGET_NAME := Path of the share folder.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure (ES=Extended settings enabled,AS=automatic subscription switched on):

Possible case of failure	Mode	Success or error behavior
WebService client adapter not started		Error. Throws acBISFXShareManagmentShareFolder fault.
No connection to BISFX server		Error. Throws acBISFXShareManagmentShareFolder fault.
Wrong/Invalid URL	ES	Error. Throws acBISFXShareManagmentShareFolder fault.
Non-existing system variable in *URL*	ES	Error. Throws acBISFXShareManagmentShareFolder fault.
Automatic-subscription settings not allowed	AS	Error. Throws acBISFXShareManagmentShareFolder fault.

Policies: none.

1.21.3.2 acBISFXShareManagmentUnshareFolder

Caption: BISFXShareManagmentUnshareFolder.

Throws Fault=acBISFXShareManagmentUnshareFolder.

The *acBISFXShareManagmentUnshareFolder* activity is an activity that unshares a folder of a SEEFX account from a single recipient. Uses the *Unsh*areFolder operation from the *ShareManagementService3* BIS FX web service (refer to chapter *1.4 ShareManagementService3 (link)*, sub chapter *1.4.2 Operations* in the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to ShareManagementService3Port web service.
Full folder path	*<FolderPath>*[2]	Mandatory. Editable text field. Default value '/My Files/'. Absolute SEE FX path to the share folder. Must follow the SEE FX Path Guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_SHARE_FOLDER]'. • System variables (page 392), e.g. '${MY_SHARE_FOLDER}'.

Field name	Value	Description
		• XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Display name	*<Text>*[3]	Mandatory. Editable text field. Empty by default. Display name of the share recipient. Supports: • Properties (page 392), e.g. '$[MY_RECIPIENT_NAME]'. • System variables (page 392), e.g. '${MY_RECIPIENT_NAME}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Home group	<GroupName>	Optional. Editable text field. Empty by default. Home group of the share recipient in case of 'User' type. Supports: • Properties (page 392), e.g. '$[REC_HOME_GROUP]'. • System variables (page 392), e.g. '${REC_HOME_GROUP}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Name		Mandatory. Editable text field. Empty by default. Name of the share recipient. Supports: • Properties (page 392), e.g. '$[MY_RECIPIENT]'. • System variables (page 392), e.g. '${MY_RECIPIENT}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
User type		Mandatory. Editable selection field. Default value 'User'. The type of the share recipient. Supports: • Properties (page 392), e.g. '$[REC_TYPE]'. • System variables (page 392), e.g. '${REC_TYPE}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
	Group	SEE FX group.
	User	Default. SEE FX user.
Unshare subfolders	True	Unshare all subfolders.
	False	Default. Do not touch subfolders.
Extended settings	True/False	Optional. Use extended settings for *WebService endpoint*. Additional field group. Refer to *Extended settings* (page 360), properties supporting type. Makes *URL*, *User name*, *Password from property*, and *Password* visible.

Field name	Value	Description
Abort on error	True/False	Refer to Common Fields (page 3).
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

[1] Reference to master data record.
[2] Arbitrary text.

Preconditions:

- WebService Client adapter started.
- ShareManagementService3 web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- TARGET_NAME := Path of the share folder.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
System variable missing	Error. Throws acBISFXShareManagmentUnshareFolder fault.

Policies: none.

1.21.3.3 acBISFXShareManagmentSetShareInfoFor

Caption: BISFX ShareManagmentSetShareInfoFor.

Throws Fault=acBISFXShareManagmentSetShareInfoFor.

The *acBISFXShareManagmentSetShareInfoFor* activity is a web service based activity that sets the ShareInfo (folder, share recipient, rights, subscription flag). Uses the *SetShareInfosForFolder* operation from the *Share ManagementService* BIS FX web service (refer to chapter *2.4 ShareManagementService (link)*, sub chapter *2.4.2 Operations* in the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to ShareManagementService3Port web service.
Full path	*<FolderPath>*[2]	Mandatory. Editable text field. Default value '/My Files/'. The absolute path of the share folder. Must follow the SEE FX Path Guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_SHARE_FOLDER]'. • System variables (page 392), e.g. '${MY_SHARE_FOLDER}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')).
Folder path		Mandatory. Editable text field. Empty by default. Supports: • Properties (page 392), e.g. '$[MY_SHARE_FOLDER]'. • System variables (page 392), e.g. '${MY_SHARE_FOLDER}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')).
Recipient		Mandatory. Editable text field. Empty by default. Name of the share recipient. Supports: • Properties (page 392), e.g. '$[RECIPIENT_NAME]'. • System variables (page 392), e.g. '${RECIPIENT_NAME}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')).
User rights		Mandatory. Editable selection field. Default value 'View'. The access rights for the share recipient for this share folder. Supports: • Properties (page 392), e.g. '$[RIGHTS]'. • System variables (page 392), e.g. '${RIGHTS}'.

Field name	Value	Description
	Delete	Allows to: • delete files or folders.
	Permit	Allows to: • create permits for this folder. • create download permits if combined with 'Read'. • create upload permits if combined with 'Write'.
	Read	Allows to: • see the content of the folder. • copy files. • download files.
	Read;VIew;	Allows to: • see content of the folder. • copy files. • download files. • view documents (saving, printing, or forwarding not allowed).
	Read;Write;Delete;View;	Includes all rights except permit related rights.
	View	Default. Allows to: • view only documents: The document can only be viewed in the brower, it is not possible to save, to print, or to forward the document.
	Write	Allows to: • copy or move own files to this folder. • edit existing files. • upload files. • create folders. • edit folders.
Is subscribed	True	Default. Subscription of this folder by the share recipient.

Field name	Value	Description
	False	No subscription of this folder by the share recpient.
Extended settings	True	Optional. Use extended settings for *WebService endpoint*. Additional field group with *URL*, *User name*, and *Password* fields. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *User name*, and *Pass word* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Absolute SEEFX folder path.
3) Valid URL to SEE FX web service.
4) Arbitrary text.

Preconditions:

- WebService Client adapter started.
- ShareManagementService3 web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
System variable missing	Error. Throws acBISFXShareManagmentSetShareInfoFor fault.

Policies: none.

1.21.3.4 acBISFXShareManagmentListShareInfoFor

Caption: BISFX ShareManagmentListShareInfoFor.

Throws Fault=acBISFXShareManagmentListShareInfoFor.

The *acBISFXShareManagmentListShareInfoFor* activity is an activity that retrieves the requested ShareInfo. Uses the *ListShare*Info*For* operation from the *ShareManagementService3* BIS FX web service (refer to chapter *1.4 ShareManagementService3 (link)*, sub chapter *1.4.2 Operations* in the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to ShareManagementService3Port web service.
Full folder path	*<FolderPath>*[2]	Mandatory. Editable text field. Default value '/My Files/'. The absolute path of the folder to be listed. Must follow the SEE FX Path Guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_SHARE_FOLDER]'. • System variables (page 392), e.g. '${MY_SHARE_FOLDER}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Extended settings	True	Optional. Use extended settings for *WebService endpoint*. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *User name*, and *Password* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Absolute SEEFX folder path.
3) Valid URL to SEE FX web service.
4) Arbitrary text.

Preconditions:

- WebService Client adapter started.
- ShareManagementService3 web service installed.

Result:

- New process attachment: contains requested ShareInfo.

Sets the following properties:

- TARGET_NAME := Path of the share folder.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Invalid System variable	Error. Throws acBISFXShareManagmentListShareInfoFor fault.

Policies: none.

1.21.4 File Transfer Service

1.21.4.1 acBISFXCreateFolder

Caption: BISFXCreateFolder.

Throws Fault=acBISFXCreateFolder.

The *acBISFXCreateFolder* activity is a web service based activity that creates a new BISFX folder. Uses the *CreateFolder3* operation of the *FileTransferService3* BISFX web service (refer to chapter *1.2 FileTransferService3* of the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to FileTransferService3Port web service.
Full folder path	*<FolderPath>*[2]	Mandatory. Editable text field. Absolute path to a BISFXFX folder. Must follow the SEEFX Path Guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_SOURCE_PATH]'. • System variables (page 392), e.g. '${MY_SOURCE_PATH}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
On folder exists		Mandatory. Editable selection field. Provides additional options for the file name of the target folder and determines what to do in case there is already an existing folder with the same name. Supports: • Properties (page 392), e.g. '$[MODE]'. • System variables (page 392), e.g. '${MODE}'.
	Auto rename	If a folder with the folder name specified in *Full folder path* already exists, a new folder is created with a new name, usually generated from the original file name and a counter.
	Fail	If a folder with the folder name specified in *Full folder path* already exists, the activity fails.
	Overwrite	Default. If a folder with the folder name specified in *Full folder path* already exists, it is overwritten. The previous folder and its content are moved to the recycle bin.
Description	*<Text>*[3]	Optional. Editable text field. The description of the folder used by BISFX. Displayed in BISFX GUI.
Extended settings	True	Use extended settings for *WebService endpoint*. Displays additional fields *URL*, *User name*, and *Password* which overwrite the settings from the *WebService endpoint*. Makes *URL*, *User name*, and *Password* visible.

Field name	Value	Description
	False	Default. Does not display additional fields.
URL		Optional. Editable text field. Empty by default. URL of the external web service. Overwrites the URL of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_BISFX_URL]'. • System variables (page 392), e.g. '${MY_BISFX_URL}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended settings* is switched on.
Sender		Optional. Editable selection field. Empty by default. Master data BIS FileExchange Personality address or Bidirectional address. Represents a technical SEE FX user. User name of HTTP basic authentication. Overwrites the HTTP authentication user name of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_BISFX_USER]'. • System variables (page 392), e.g. '${MY_BISFX_USER}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended settings* is switched on.
Password from property	True	Uses clear text password without encryption. Makes *Encryption password clear text* visible.
	False	Default. Uses password field and password encryption. Makes *Password* visible.
Password	*<Password>*[4]	Optional. Editable password field. Empty by default. Password of HTTP basic authentication. Overwrites the HTTP authentication password of the *WebService endpoint*. System variables and properties are not supported. Visible when *Extended settings* is switched on and *Password from property* is switched off.
Encryption password clear text	*<Password>*[4]	Optional. Editable text field. Empty by default. Password of HTTP basic authentication. Overwrites the HTTP authentication password of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_BISFX_PW]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). System variables are not allowed.

Field name	Value	Description
		Visible when *Extended settings* and *Password from property* are switched on.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Absolute BISFX folder path.
3) Arbitrary text.
4) Secure password.

Preconditions:

- WebService Client adapter started.
- FileTransferService3 web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- TARGET_NAME := Value from *Full folder path*.
- STATUS := 'SUCCESS', 'ERROR'.
- RESULT := '100', '204'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXCreateFolder fault.
WebService client adapter not started	Error. Throws acBISFXCreateFolder fault.
No connection to BISFX server	Error. Throws acBISFXCreateFolder fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXCreateFolder fault.
Unsufficient rights for user	Error. Throws acBISFXCreateFolder fault.
System variable not existing	Error. Throws acBISFXCreateFolder fault.
Folder with same name already exists when *On folder exists* is set to 'Fail'	Error. Throws acBISFXCreateFolder fault.

Policies: none.

1.21.4.2 acBISFXDeleteFile

Caption: BISFXDeleteFile.

Throws Fault=acBISFXDeleteFile.

The *acBISFXDeleteFile* activity is a web service based activity that deletes a file of a BISFX account. Uses the *ListContent3* and *Delete* operations from the *FileTransferService3* BISFX web service (refer to chapter *1.2 FileTransferService3* of the *BIS FileExchange WebService API HowTo* manual for more details).

I The *Delete* operation of the *FileTransferService3* BISFX web service is deprecated. It is recommended to use *acBISFXRemove* (page 349) activity instead.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to FileTransferService3Port web service.
Path	<FilePath>[2]	Mandatory. Editable text field. Default value '/My Files/'. Absolute path to a BISFX file. Must follow the SEE FX Paths guidelines (page 398). Supports: • Properties (page 392), e.g. '$[TARGET_PATH]'. • System variables (page 392), e.g. '${TARGET_PATH}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Operation		Mandatory. Non-editable selection field. Supported deletion modes.
	Delete permanently	Default. Delete source file permanently.
	Move to recycle bin	Move source file to BISFX recycle bin. Does not work if the *Pat h* starts with '/Recycle Bin/', files from the recycle bin are always deleted permanently.
Ignore file not found	True	If the file is not existing continue the process.
	False	Default. If the file is not existing throw an error. Resume or abort the process due to *Abort on error* switch.
Extended settings	True/False	Optional. Use extended settings for *WebService endpoint*. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *User name*, and *Password* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Default value 'False'. Refer to Common Fields (page 3).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Absolute BISFX path.
3) Arbitrary text.

Preconditions:

- FileTransferService3 web service installed.
- Web Services Client adapter started.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- TARGET_PATH := The path to the folder of the deleted file.
- TARGET_NAME := The name of the deleted file.

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Ignore file not found	Success or error behavior
File not found	False	Error. Throws acBISFXDeleteFile fault.
File not found	True	Success. Process continues with next activitiy.
Non-existing folder instead of file		Error. Throws acBISFXDeleteFile fault.
Insufficient rights		Error. Throws acBISFXDeleteFile fault.

Policies: none.

1.21.4.3 acBISFXDeleteItem

Caption: BISFXDeleteItem.

Throws Fault=acBISFXDeleteItem.

The *acBISFXDeleteItem* activity is a web service based activity that deletes a file or a folder of a BISFX account. Uses the *DeleteItem* operation from the *FileTransferService3* BISFX web service (refer to chapter *1.2 FileTransferService3* of the *BIS FileExchange WebService API HowTo* manual for more details).

The *Delete* operation of the *FileTransferService3* BISFX web service is deprecated. It is recommended to use *acBISFXRemove* (page 349) activity instead.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to FileTransferService3Port web service.
Full item path	<FilePath>[4]	Mandatory. Editable text field. Default value '/My Files/'. Absolute path to a BISFX file or folder. Must follow the SEE FX Paths guidelines (page 398). Supports: • Properties (page 392), e.g. '$[TARGET_PATH]'. • System variables (page 392), e.g. '${TARGET_PATH}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). If the *Full item path* is a folder path all files inside are also deleted/moved to the recycle bin.
Delete item immediately	True	Default. Item is deleted permanently.
	False	Item is moved to recycle bin.
Extended settings	True/False	Optional. Use extended settings for *WebService endpoint*. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *User name*, and *Pass word* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Default value 'False'. Refer to Common Fields (page 3).
Log message	*<Text>*[5]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Valid URL to BISFX web service.
3) Secure password.
4) Absolute SEE FX path.
3) Arbitrary text.

Preconditions:

- FileTransferService3 web service installed.
- Web Services Client adapter started.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- TARGET_PATH := The path to the folder of the deleted file.
- TARGET_NAME := The name of the deleted file.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Ignore file not found	Success or error behavior
File not found	False	Error. Throws acBISFXDeleteItem fault.
File not found	True	Success. Process continues with next activitiy.
Non-existing folder instead of file		Error. Throws acBISFXDeleteItem fault.
Insufficient rights		Error. Throws acBISFXDeleteItem fault.

Policies: none.

1.21.4.4 acBISFXRemove

Caption: BISFXDeleteItem.

Throws Fault=acBISFXRemove.

The *acBISFXRemove* activity is a web service based activity that deletes a file or a folder of a BISFX account. Uses the *Remove* operation from the *FileTransferService3* BISFX web service (refer to chapter *1.2 FileTransferService3* of the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to FileTransferService3Port web service.
Extended Settings	True/False	Optional. Use extended settings for *WebService endpoint*. Additional field group. Refer to *Extended settings* (page 360), standard type. Makes *URL*, *User name*, and *Pass word* visible.
Full item path	<FilePath>[2]	Mandatory. Editable text field. Default value '/My Files/'. Absolute path to a BISFX file or folder. Must follow the SEEFX Paths guidelines (page 398). Supports: • Properties (page 392), e.g. '$[TARGET_PATH]'. • System variables (page 392), e.g. '${TARGET_PATH}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). If the *Full item path* is a folder path all files inside are also deleted/moved to the recycle bin.
Delete item immediately	True	Default. Item is deleted permanently.
	False	Item is moved to recycle bin.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Default value 'False'. Refer to Common Fields (page 3).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Absolute BISFX path.
3) Arbitrary text.

Preconditions:

- FileTransferService3 web service installed.

- Web Services Client adapter started.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- TARGET_PATH := The path to the folder of the deleted file.
- TARGET_NAME := The name of the deleted file.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Ignore file not found	Success or error behavior
File not found	False	Error. Throws acBISFXRemove fault.
File not found	True	Success. Process continues with next activitiy.
Non-existing folder instead of file		Error. Throws acBISFXRemove fault.
Insufficient rights		Error. Throws acBISFXRemove fault.

Policies: none.

1.21.4.5 acBISFXGetValueFor

Caption: BISFXGetValueFor .

Throws Fault=acBISFXGetValueFor.

The *acBISFXGetValueFor* activity is a web service based activity that retrieves the value for the specified property in BISFX. Uses the *GetValueFor* operation from the *FileTransferService3* BISFX web service (refer to chapter *1.2 FileTransferService3* in the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to FileTransferService3Port web service.
Path	*<FolderPath>*[2]	Mandatory. Editable text field. Empty by default. Absolute path to a SEE FX folder. Must follow the SEE FX Path Guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_PATH]'. • System variables (page 392), e.g. '${MY_PATH}'.
Name	*<PropName>*[3]	Mandatory. Editable text field. Empty by default. Name of the property whose value is to be retrieved. Supports: • Properties (page 392), e.g. '$[PROP_NAME]'. • System variables (page 392), e.g. '${PROP_NAME}'.
Extended settings	True/False	Optional. Use extended settings for *WebService endpoint*. Additional field group. Default fields *Login with Token*, *URL*, *U ser name*, *Password* and *Login Token* are described in Extended settings with Token (page 305). Makes *Login with Token*, *URL*, *User name*, *Password* and *Login token* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Absolute BISFX folder path.
3) Valid BISFX property name.
4) Arbitrary text.

Preconditions:

- WebService Client adapter started.
- FileTransferService3 web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- BISFX_GET_VALUE := the value of the BISFX property.
- STATUS := 'SUCCESS', 'ERROR'.
- RESULT := '100', '204'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationCreateUser fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationCreateUser fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationCreateUser fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationCreateUser fault.
Unsufficient rights for user	Error. Throws acBISFXAdministrationCreateUser fault.
System variable not existing	Error. Throws acBISFXAdministrationCreateUser fault.

Policies: none.

1.21.4.6 acBISFXSetValueFor

Caption: BISFXSetValueFor .

Throws Fault=acBISFXSetValueFor.

The *acBISFXSetValueFor* activity is a web service based activity that sets the new value for the specified BIS FX property. Uses the *SetValueFor* operation from the *FileTransferService3* BISFX web service (refer to chapter *1.2 FileTransferService3* in the *BIS FileExchange WebService API HowTo* manual for more details).

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to FileTransferService3Port web service.
Path	*<FolderPath>*[2]	Mandatory. Editable text field. Empty by default. Absolute path to a BISFX folder. Must follow the SEE FX Path Guidelines (page 398). Supports: • Properties (page 392), e.g. '$[MY_PATH]'. • System variables (page 392), e.g. '${MY_PATH}'.
Name	*<PropName>*[3]	Mandatory. Editable text field. Empty by default. Name of the BISFX property whose value is to be retrieved. Supports: • Properties (page 392), e.g. '$[PROP_NAME]'. • System variables (page 392), e.g. '${PROP_NAME}'.
Read only	True	Default. Sets the ReadOnly attribute of the BISFX property to 'TRUE'.
	False	Sets the ReadOnly attribute of the BISFX property to 'FALSE'.
Value		Mandatory. Editable text field. Empty by default. The value of the BISFX property.
Recursive	True	Apply the BISFX property recursively to the complete folder tree.

Field name	Value	Description
	False	Default. Apply the property to the specified folder only.
Extended settings	True/False	Optional. Use extended settings for *WebService endpoint*. Additional field group. Default fields *Login with Token*, *URL*, *U ser name*, *Password* and *Login Token* are described in Extended settings with Token (page 305). Makes *Login with Token*, *URL*, *User name*, *Password* and *Login token* visible.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Absolute BISFX folder path.
3) Valid BISFX property name.
4) Arbitrary text.

Preconditions:

- WebService Client adapter started.
- FileTransferService3 web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- BISFX_GET_VALUE := the value of the BISFX property.
- STATUS := 'SUCCESS', 'ERROR'.
- RESULT := '100', '204'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong configuration of web service	Error. Throws acBISFXAdministrationCreateUser fault.
WebService client adapter not started	Error. Throws acBISFXAdministrationCreateUser fault.
No connection to BISFX server	Error. Throws acBISFXAdministrationCreateUser fault.
Invalid URL in *Extended Settings*	Error. Throws acBISFXAdministrationCreateUser fault.
Unsufficient rights for user	Error. Throws acBISFXAdministrationCreateUser fault.
System variable not existing	Error. Throws acBISFXAdministrationCreateUser fault.

Policies: none.

1.21.5 Secure Client Login Service

1.21.5.1 acBISFXSecureClientLogin

Caption: BISFXSecureClientLogin.

Throws Fault=acBISFXSecureClientLogin.

The *acBISFXSecureClientLogin* activity is a web service based activity that performs a client login at a BISFX server. Uses the *UsernamePasswordLogin* and the *TrustedLogin* operations from the *SecureClientLoginService* BISFX web service.

Mode of operation:
Performs a client login at a BISFX server and returns an authentication token which can be used by succeeding activities. Currently, *acSEEFXDownloadFile* (page 283), *acSEEFXUploadFile2* (page 298) , acBISFXGetValueFor (page 351), and acBISFXSetValueFor (page 353) activities are offering an authentication with these tokens. Be sure to perform a proper logout with *acBISFXSecureClientLogout* (page 358) activity afterwards.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to SecureClientLoginServicePort web service.
Authentication type		Mandatory. Non-editable selection field. Supported login types for BISFX server.
	TRUSTED	Default. Uses the *TrustedLogin* operation.
	USRPWD	Uses the *UsernamePasswordLogin* operation. Makes *Login password* visible.
Login name		Mandatory. Editable text field. Default value empty. BISFX login name. Supports: • Properties (page 392), e.g. '$[LOGIN_NAME]'. • System variables (page 392), e.g. '${MY_LOGIN_NAME}'.
Login password	*<Password>*[2]	Mandatory. Editable password field. Empty by default. Password for user from *Login name*.

Field name	Value	Description
		Visible when *Authentication type* is set to 'USRPWD'.
Extended setting	True	Optional. Extended setting for *WebService endpoint*. Additional field group. Displays additional field *URL*. Makes *URL* visible.
	False	Default. No additional fields displayed.
URL	*<URL>*[3]	Optional. Editable text field. Empty by default. Example: 'http://10.11.23.54:8080/portal-seefx/ws/FileTransferService3'. URL of the external web service. Overwrites the URL of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_BISFX_URL]'. • System variables (page 392), e.g. '${MY_BISFX_URL}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended settings* is switched on.
Queue	*<Text>*[4]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Reference to master data record.
2) Secure password.
3) Valid URL to BISFX web service.
4) Arbitrary text.

Preconditions:

- WebService Client adapter started.
- SecureClientLoginService web service installed.

Result:

- Current process attachment: not used and not changed.
- Authentication token.

Sets the following properties:

- SECURE_CLIENT_TOKEN := Authentication token returned by login operation.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
WebService client adapter not started		Error. Throws acBISFXSecureClientLogin fault.
No connection to BISFX server		Error. Throws acBISFXSecureClientLogin fault.
Wrong/Invalid URL	Extended settings enabled	Error. Throws acBISFXSecureClientLogin fault.
Non-existing system variable in *U RL*	Extended settings enabled	Error. Throws acBISFXSecureClientLogin fault.
Missing certificate/s		Error. Throws acBISFXSecureClientLogin fault.
Invalid/expired certificate/s		Error. Throws acBISFXSecureClientLogin fault.
Invalid user name for login		Error. Throws acBISFXSecureClientLogin fault.
Invalid password	USRPWD	Error. Throws acBISFXSecureClientLogin fault.

Policies: none.

1.21.5.2 acBISFXSecureClientLogout

Caption: BISFXSecureClientLogout.

Throws Fault=acBISFXSecureClientLogout.

The *acBISFXSecureClientLogout* activity is a web service based activity that performs a client logout at a BISFX server. Uses the *Logout* operation from the *SecureClientLoginService* BISFX web service.

Mode of operation:
Performs a client logout at a BISFX server with a session token generally received by a preceding *acBISFXSecureClientLogin* (page 356) activity. There is no response from the BISFX server.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non-editable selection field. Provides master data external web services. Restricted to SecureClientLoginServicePort web service.
Token	*<Token>*[2]	Mandatory. Editable text field. Default value '$[SECURE_CLIENT_TOKEN]'. BIS FX authentication token, generally returned by *acBISFX SecureClientLogin* (page 356) activity. Supports: • Properties (page 392), e.g. '$[AUTH_TOKEN]'. • System variables (page 392), e.g. '${AUTH_TOKEN}'.
Extended settings	True	Optional. Extended settings for *WebService endpoint*. Additional field group. Displays additional field *URL*. Makes *URL* visible.
	False	Default. No additional fields displayed.
URL	*<URL>*[3]	Optional. Editable text field. Empty by default. Example: 'http://10.11.23.54:8080/portal-seefx/ws/FileTransferService3'. URL of the external web service. Overwrites the URL of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_SEEFX_URL]'. • System variables (page 392), e.g. '${MY_SEEFX_URL}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible when *Extended settings* is switched on.
Queue	*<Text>*[4]	Mandatory. Refer to Common Fields.
Abort on error	True/False	Refer to Common Fields.
Log message	*<Text>*[4]	Optional. Refer to Common Fields.

1) Reference to master data record.
2) Valid SEEFX server session token.
3) Valid URL to BISFX web service.

4) Arbitrary text.

Preconditions:

- WebService Client adapter started.
- SecureClientLoginService web service installed.

Result:

- Current process attachment: not used and not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Mode	Success or error behavior
WebService client adapter not started		Error. Throws acBISFXSecureClientLogout fault.
No connection to BISFX server		Error. Throws acBISFXSecureClientLogout fault.
Wrong/Invalid *URL*	Extended settings enabled	Error. Throws acBISFXSecureClientLogout fault.
Non-existing system variable in *U RL*		Error. Throws acBISFXSecureClientLogout fault.
Missing certificate/s		Error. Throws acBISFXSecureClientLogout fault.
Invalid/expired certificate/s		Error. Throws acBISFXSecureClientLogout fault.
Invalid/expired token		Error. Throws acBISFXSecureClientLogout fault.
Non-existing system variable in *To ken*		Error. Throws acBISFXSecureClientLogout fault.
Non-existing property in *Token*		Error. Throws acBISFXSecureClientLogout fault.

Policies: none.

1.21.6 Extended Settings

The *Extended settings* field group is used in all BIS FX related activities that are based on a BISFX web service. These activities refer to this section. Two types of this field group are supported, the standard type and the properties supporting type, the difference is in the password handling (see below).

The first fields are identical:

Field name	Value	Description
Extended settings	True	Use extended settings. Displays additional fields *URL*, *Sender*, and *P assword* which overwrite the settings from the *WebService endpoint*.

Field name	Value	Description
		Makes *URL*, *Sender*, and *Password* visible.
	False	Default. No additional fields displayed.
URL	*<URL>*[1]	Optional. Editable text field. Empty by default. Example: 'http://10.11.23.54:8080/portal-seefx/ws/FileTransferService3'. URL of the external web service. Overwrites the URL of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_SEEFX_URL]'. • System variables (page 392), e.g. '${MY_SEEFX_URL}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended settings* is switched on.
User name	*<Ref>*[2]	Optional. Editable selection field. Masterdata BIS FileExchane addresses. User name of HTTP basic authentication, represents a BISFX user. Overwrites the HTTP authentication user name of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_SEEFX_USER]'. • System variables (page 392), e.g. '${MY_SEEFX_USER}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Extended settings* is switched on.
...		Password part. Either standard type or properties supporting type. Refer to the descriptions below. Visible when *Extended settings* is switched on.

1) Valid URL to BISFX web service.
2) Reference to master data.

Extended Settings with Standard Password Field

This field group is visible when *Extended settings* is switched on.

Field name	Value	Description
Password	*<Password>*[3]	Optional. Editable password field. Password of HTTP basic authentication. Overwrites the HTTP authentication password of the *WebService endpoint*. System variables and properties are not supported.

1) Secure password.

Extended Settings with Properties Supporting Password Field

This field group is visible when *Extended settings* is switched on.

Field name	Value	Description
Password from property	True	Uses clear text password without encryption. Be aware that the password is stored and transmitted in plain text. Makes *Encryption password clear text* visible.
	False	Default. Uses password field and password encryption. Makes *Password* visible.
Password	*<Password>*[1]	Optional. Editable password field. Password of HTTP basic authentication. Overwrites the HTTP authentication password of the *WebService endpoint*. The password is stored and transmitted in an encrypted way. System variables and properties are not supported. Visible when *Password from property* is switched off.
Encryption password clear text	*<Password>*[1]	Optional. Editable text field that displays the password in plain text. Empty by default. Password of HTTP basic authentication. Overwrites the HTTP authentication password of the *WebService endpoint*. The password is stored and transmitted in plain text. Supports: • Properties (page 392), e.g. '$[MY_SEEFX_PW]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). System variables are not allowed. Visible when *Password from property* are switched on.

1) Secure password.

1.21.7 Extended Settings with Token

The Extended settings field group is used in all SEEFX related activities that are based on a SEEFX web service. This is a special type providing an additional login mode using tokens which currently is supported by *acSEEFXDownloadFile* (page 283), *acSEEFXUploadFile2* (page 298), *acBISFXGetValueFor* (page 351), and *acBISFXSetValueFor* (page 353) activities.They refer to this section.

Field name	Value	Description
Extended settings	True	Use extended settings. Displays additional fields *Login with token*, *URL*, *Sender*, *Password*, and *Login token*. *URL*, *Sender*, and *Password* overwrite the settings from the *WebService endpoint*. Makes *Login with token*, *URL*, *Se nder*, *Password*, and *Login token* visible.

Field name	Value	Description
	False	Default. No additional fields displayed.
Login with token		Switches between login modes. Visible when *Extended settings* is switched on.
	True	Use login token instead of user and password. Makes *Login token* visible.

Field name	Value	Description
	False	Use user and password for login. Makes *User name* and *Password* visible.
URL	*<URL>*[1]	Optional. Editable text field. Empty by default. Example: 'http://10.11.23.54:8080/portal-seefx/ws/FileTransferService3'. URL of the external web service. Overwrites the URL of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_SEEFX_URL]'. • System variables (page 392), e.g. '${MY_SEEFX_URL}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible when *Extended settings* is switched on.
User name	*<Ref>*[2]	Optional. Editable text field. Empty by default. User name of HTTP basic authentication, represents a SEEFX user. Overwrites the HTTP authentication user name of the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[MY_SEEFX_USER]'. • System variables (page 392), e.g. '${MY_SEEFX_USER}'. • XPath expressions (page 392), e.g. $(upper-case('filename')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('0000000000')). Visible when *Extended settings* is switched on and *Login with token* is switched off.
Password	*<Password>*[3]	Optional. Editable password field. Password of HTTP basic authentication. Overwrites the HTTP authentication password of the *WebService endpoint*. System variables and properties are not supported. Visible when *Extended settings* is switched on and *Login with token* is switched off.
Login token	*<Token>*[4]	Optional. Editable text field. Empty by default. Valid session token from SEEFX server

1) Valid URL to SEEFX web service.
2) Reference to master data.
3) Secure password.
4) SEEFX server login token.

1.22 Webservice Activities

This section describes web service related activities.

1.22.1 acGenericServiceReply

Caption: Reply to GenericService Call.

Throws Fault=acGenericServiceReply.

The *acGenericServiceReply* activity is an activity that creates a reply message for an incoming web service call, for instance by *acGenericWS* (page 368) activity, with one message part.

Mode of operation:
The *acGenericServiceReply* activity has to be used as a part of the *MFT-WebServiceEvent* process (refer to chapter *MFT-WebServiceEvent* in the *MFT Standard Solution - Adapters* manual for more details), i.e. the Criteria *From adapter* field has to set to '*W*ebservice'. The message part that is to be sent as part of the reply message is taken from either the current process attachment or a bookmarked attachment.

Only messages with one part of Document style model and literal use model only are supported. Several headers are supported but each header may contain only one part.

Field name	Value	Description
Part name	*<PartName>*[1]	Mandatory. Editable text field. Empty by default. Name of message part from the underlying WSDL file. System variables and properties are not supported.
Attachment (bookmark)		Mandatory. Editable selection field. The selected attachment is used as the message part of the reply message and should contain the web service answer according to the underlying WSDL file. Supports: • Properties (page 392), e.g. '$[MY_ATTACHMENT_NAME]'. • System variables (page 392), e.g. '${MY_ATTACHMENT_NAME}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
	<current payload>	Default. Use the current process attachment.
	<BookmarkName>[2]	Name of an existing bookmarked attachment.
Header provider		Optional. Editable selection field. Provides supported modes of creating header messages for the web service request. Note that each header can have only one part. System variables and properties are not supported.

Field name	Value	Description
	No header	Default. Do not create a header.
	Mapping	Creates one or several headers using a mapping. Makes *Header mapping* visible.
	Bookmark	Creates exactly one header from the content of a bookmarked attachment. Makes *Header bookmark* visible.
	Script	Creates one or several headers using a beanshell script. Makes *Header script* visible.
Header mapping	*<Ref>*[3]	Optional. Editable selection field. Provides available mappings. The mapping gets all current properties as input properties. It can create one or multiple headers: • Single header: Provides the single header message as the main output stream of the mapping together with two mandatory output properties: $[MESSAGE_NAME] := The fully qualified message QName[6]. $[PART_NAME] := The name of the part for the header message[1]. • Multiple headers: Each header is returned as an additional output stream. Each additional output stream is expected to have a name according to '<MessageQName>.<PartName>' [6,1], for instance '{http://uri.seeburger.com/header}ServiceHeader.header'. Supports: • Properties (page 392), e.g. '$[MY_MAPPING]'. • System variables (page 392), e.g. '${MY_MAPPING}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Header provider* is set to 'Mapping'.
Header bookmark	*<BookmarkName>*[2]	Optional. Editable text field. Case sensitive name of an existing bookmarked attachment. The bookmark name is expected to be a name according to '<MessageQName>.<PartName>' [6,1], for instance '{http://uri.seeburger.com/header}ServiceHeader.header'. Supports: • Properties (page 392), e.g. '$[MY_BOOKMARK_NAME]'. • System variables (page 392), e.g. '${MY_BOOKMARK_NAME}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Header provider* is set to 'Bookmark'.

Field name	Value	Description
Header script	*<ScriptName>*[4]	Optional. Editable text field. Name of beanshell script that creates one or several headers. The script gets all current properties as input properties. It returns one or several attachments, one attachment for each created header together with the output property '<AttachmentRefID>=<MessageQName>.<PartName>'[6,1], for instance 'f370f550-84a7-11e6-92af-cff30a006994={http://uri.seeburger.com/header}ServiceHeader.header'. Supports: • Properties (page 392), e.g. '$[MY_SCRIPT]'. • System variables (page 392), e.g. '${MY_SCRIPT}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Header provider* is set to 'Script'. Refer to *Creating a BSH Script* (page 410) for details about script implementation.
Log message	*<Text>*[5]	Optional. Refer to Common Fields (page 1).

1) Message part name from underlying WSDL.
2) Valid name of existing bookmarked attachment.
3) Reference to master data record.
4) Name of beanshell script from repository including file extension.
5) Arbitrary text.
6) Fully qualified message QName. Consists of namespace and a local part.

Preconditions:

- WebService controller started.
- The Criteria must match the *MFT-WebServiceEvent* event, i.e. the *From adapter* field has to set to 'Webservice'.
- The attachment must contain the message part, either the current process attachment or a bookmarked attachment.

Result:

- Created reply message sent to calling web service.

Sets the following properties:

- GENERICSERVICE_REPLY_HEADER := header of the reply message, Base 64 encoded.
- GENERICSERVICE_REPLY_MESSAGE := part of the reply message, Base 64 encoded.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Non-existing bookmark in *Attachment (bookmark)*	Error. Throws acGenericServiceReply fault.
Non-existing mapping or invalid mapping name in *Header mapping*	Error. Throws acGenericServiceReply fault.
Non-existing bookmark or invalid bookmark name in *Header bookmark*	Error. Throws acGenericServiceReply fault.
Non-existing BSH script or invalid script name in *Header script*	Error. Throws acGenericServiceReply fault.

Policies: none.

1.22.2 acGenericWS

Caption: Call WebService endpoint.

Throws Fault=acGenericWS.

The *acGenericWS* activity is an activity that calls an external web service in a generic way. The web service call can be synchronous or asynchronous.

Only messages with one part of Document style model and literal use model only are supported. Several headers are supported but each header may contain only one part.

Only HTTP authentication from *WebService endpoint* can be overwritten by using the *Extended settings*. Overwriting TLS authentication, WSS authentication and signing/encrypting with certificates is not supported.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non editable selection field. External web service endpoints from master data *Consumed services* > *Web services*.
Operation	*<Ref>*[1]	Mandatory. Non editable selection field. Empty by default. Shows all operations provided by the selected *WebService endpoint*.
Part name	*<PartName>*[2]	Mandatory. Editable text field. Empty by default. Name of message part from the underlying WSDL file for the selected *Operation*. Refer to "Finding the part name" below for details about its location. System variables and properties are not supported.
Async call	True	Uses asynchronous mode for calling the web service. Does not wait for a response and continues immediately with the next activity.
	False	Default. Uses synchronous mode for calling the web service. Waits for the response of the web service.
Payload envelop script	*<ScriptName>*[3]	Optional. Editable text field. Empty by default. Name of beanshell script that wraps the attachment in an (outer) XML structure if necessary. Refer to "Meeting the required message structure" for details about the correct message structure for a web service request. System variables and properties are not supported. Refer to *Creating a BSH Script* (page 410) for more details about script implementation.

Field name	Value	Description
Header provider		Optional. Editable selection field. Provides supported modes of creating header messages for the web service request. Note that each header can have only one part. System variables and properties are not supported.
	No header	Default. Do not create a header.
	Mapping	Creates one or several headers using a mapping. Makes *Header mapping* visible.
	Bookmark	Creates exactly one header from the bookmarked attachment. Makes *Header bookmark* visible.
	Script	Create one or several headers using a beanshell script. Makes *Header script* visible.
Header mapping	*<Ref>*[1]	Optional. Editable selection field. Provides available mappings. The mapping gets all current properties as input properties. It can create one or multiple headers: • Single header: Provides the single header message as the main output stream of the mapping together with two mandatory output properties: $[MESSAGE_NAME] := The fully qualified message QName[9]. $[PART_NAME] := The name of the part for the header message[2]. • Multiple headers: Each header is returned as an additional output stream. Each additional output stream is expected to have a name according to '<MessageQName>.<PartName>' [9,2], for instance '{http://uri.seeburger.com/header}ServiceHeader.header'. System variable and properties are not supported. Visible when *Header provider* is set to 'Mapping'.
Header bookmark	*<BookmarkName>*[4]	Optional. Editable text field. Case sensitive name of an existing bookmarked attachment. The bookmark name is expected to be a name according to '<MessageQName> .<PartName>' [9,2], for instance '{http://uri.seeburger.com/header}ServiceHeader.header'. Supports: • Properties (page 392), e.g. '$[MY_BOOKMARK_NAME]'. • System variables (page 392), e.g. '${MY_BOOKMARK_NAME}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Header provider* is set to 'Bookmark'.
Header script	*<ScriptName>*[3]	Optional. Editable text field. Name of beanshell script that creates one or several headers. The script gets all current properties as input properties. It returns one or several attachments, one attachment for each created header together with the output

Field name	Value	Description
		property '<AttachmentRefID>=<MessageQName>.<PartName>'[9,2], for instance 'f370f550-84a7-11e6-92af-cff30a006994={http://uri.seeburger.com/header}ServiceHeader.header'. System variables and properties are not supported. Visible when *Header provider* is set to 'Script'. Refer to *Creating a BSH Script* (page 410) for more details about script implementation.
Extended settings	True	Use extended settings. Display *URL*, *User name*, and *Password* which overwrite the settings from the *WebService endpoint*.
	False	Default. No extended settings, no additional fields visible.
URL	*<URL>*[5]	Optional. Editable text field. URL of the external web service. Overwrites the URL of the *WebService endpoint*. Properties and system variables are not supported. Visible when *Extended settings* is switched on.
User name	<UserName>	Optional. Editable text field. User name of HTTP basic authentication. Overwrites the HTTP authentication user name of the *WebService endpoint*. Properties and system variables are not supported. Visible when *Extended settings* is switched on.
Password	*<Password>*[6]	Optional. Editable password field. Password of HTTP basic authentication. Overwrites the HTTP authentication password of the *WebService endpoint*. Properties and system variables are not supported. Visible when *Extended settings* is switched on.
Security token	True	Add security-related message information. Makes *Token value type* and *Token* visible.
	False	Default. No additional fields visible.
Token value type		Mandatory. Editable text field. Empty by default. The type of the security token, e.g. X.509 certificates or Kerberos tickets. Supports: • Properties (page 392), e.g. '$[TOKEN_TYPE]'. • System variables (page 392), e.g. '${TOKEN_TYPE}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Security token* is switched on.
Token	*<Binary>*[7]	Mandatory. Editable text field. Empty by default. Binary encoded security token. Supports:

Field name	Value	Description
		• Properties (page 392), e.g. '$[TOKEN]'. • System variables (page 392), e.g. '${TOKEN}'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). Visible when *Security token* is switched on.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Abort on error	True/False	Refer to Common Fields (page 3).
Log message	*<Text>*[8]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Message part name from underlying WSDL.
3) Name of beanshell script from repository including file extension.
4) Arbitrary case-sensitive name except invalid characters ", &, <. Name of existing bookmarked attachment.
5) Valid URL of external web service.
6) Secure password.
7) Binary encoded value.
8) Arbitrary text.
9) Fully qualified message QName. Consists of namespace and a local part.

Preconditions:

- Current process attachment: XML document.

Result:

- Synchronous mode: New process attachment in XML format, contains the message part of the response from the web service.

The reply message may contain only one part. Several parts are not supported.

- Synchronous mode: Bookmarked attachments for each reply header. Bookmark name format is '<MessageQName>.<PartName>'.

Sets the following properties:

In asynchronous mode:

- SENDING_TIME := sending time, format 'yyyyMMddHHmmssSSS'.
- STATUS := 'SUCCESS'.
- RESULT := '100'.

In synchronous mode:

- SENDING_TIME := sending time, format 'yyyyMMddHHmmssSSS'.
- POT_TYPE := port type.
- OPERATION := operation.
- TARGET_NAME := target name.
- COUNT_HEADERS := number of headers.
- STATUS := 'SUCCESS'.

- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Header provider	Success or error behavior
Invalid file format for attachment		Error. Throws acGenericWS fault.
Missing or wrong authorization		Error. Throws acGenericWS fault.
Invalid message part, does not match WSDL		Error. Throws acGenericWS fault.
Invalid bookmark name, does not match WSDL	Bookmark	Error.
Not existing bookmark	Bookmark	Error.
Not existing system variable (holding bookmark name)	Bookmark	Error. Throws acGenericWS fault.
Not existing mapping	Mapping	Error. Throws acGenericWS fault.
Not existing script	Script	Error.

Policies: none.

Finding the part name

The name of the message part is mandatory and requires to inspect the WSDL file for the appropriate name. How to find it will be shown using SEE FX FileTransferService3 WSDL file as an example.

1. Look for

```
<wsdl:binding
…
```

2. Inside this element, you have to look for the operation name entered in the *Operation* field above. This example uses the “Upload” operation and it looks like that:

```
<wsdl:operation name="Upload">

<soap:operation soapAction="" style="document" />

<wsdl:input name="Upload">

<soap:body use="literal" />

</wsdl:input>

…
```

3. This "operation" element contains a child element called "input":

```
<wsdl:input name="Upload">
```

This "input" element defines the name of the message that is used for the operation, "Upload" in this example.

4. Now, look for a" message" element with the name "Upload":

```
<wsdl:message name="Upload">

<wsdl:part element="tns:Upload" name="parameters">

</wsdl:part>
```

```
</wsdl:message>
```

5. This "message" element contains a child element called "part" with its name, the desired part name, "parameters" in this example.

Meeting the required message structure

The current process attachment that is sent with the web service request needs to match the message structure required by the selected operation. There are some cases where it might be necessary that the original attachment can be wrapped in an outer xml structure.

An example is the interface between BIS and SEEBURGER SAP InvoiceConsole. The example operation is "initAndSupplyData", the part name is "parameters". It is a simple XML message with an embedded base 64 zip string. The appropriate zip file is normally created during standard processing and assigned to the process as current process attachment. Wrapping this zip file into the expected message structure can be done like follows:

1. Use a mapping in order to create the request message:
   Use the acConverter (page 63) activity to wrap the zip file before the *acGenericServiceWS* activity is called.
2. Use a beanshell script in order to create the request message inside the *acGenericServiceWS* activity:
   The name of the beanshell script can be set using the *Payload envelop script* field.
   For SEEBURGER SAP Invoice Console a standard script with name “CreateDocSuiteEnvelop.bsh” is provided with S-18000-WS plugin. Refer to *Creating a BSH Script* (page 410) for more details about script implementation.

The SoapUI tool can be used to load the WSDL file and get an example XML of the required message.

1.22.3 acUploadFile

Caption: Call UploadFile endpoint.

Throws Fault=acUploadFile.

The *acUploadFile* activity is a single file activity that uploads the current process attachment using the specified *WebService endpoint*. This web service request can be performed synchronously or asynchronously and triggers an *MFT-UploadFileEvent* (refer to chapter *MFT-UploadFileEvent* in the *MFT Standard Solution - Adapters* manual for more details) on the target system of the web service endpoint.

Only HTTP authentication is supported. WSS authentication and signing/encrypting with certificates is not supported.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non editable selection field. External web service endpoints from master data with 'MFTServiceServicePort' port name.
Send asynchronous	True	Performs the upload in an asynchronous way. Continues immediately with the next activity. Triggers the asynchronous version of the *MFT-UploadFileEvent*.
	False	Performs the upload in a synchronous way. Waits for the response. Triggers the synchronous version of the *MFT-UploadFileEvent* sending a response containing the process attachment and the TRACK_ID of the new process.
Subject	*<Text>*[2]	Optional. Editable text field. Request subject, description of requested operation. Mapped to *From subject* of triggered *MFT-UploadFileEvent*. Supports:

Field name	Value	Description
		• System variables (page 392), e.g. '${MY_SUBJECT}'. • Properties (page 392), e.g. '$[MY_SUBJECT]'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
From Path	*<Text>*[2]	Optional. Editable text field. Description of attachment. Mapped to *From path* of triggered *MFT-UploadFileEvent*. Supports: • System variables (page 392), e.g. '${MY_PATH}'. • Properties (page 392), e.g. '$[MY_PATH]'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
From name	*<Text>*[2]	Optional. Editable text field. Name of request sender. Mapped to *From name* of triggered *MFT-UploadFileEvent*. Supports: • System variables (page 392), e.g. '${SENDER_NAME}'. • Properties (page 392), e.g. '$[SENDER_NAME]'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
From ID	*<Text>*[2]	Optional. Editable text field. Sender ID, should be unique. Mapped to *From ID* of triggered *MFT-UploadFileEvent*. Supports: • System variables (page 392), e.g. '${SENDER_ID}'. • Properties (page 392), e.g. '$[SENDER_ID]'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Properties (key=value;...)	*<KVList>*[3]	Optional. Editable text field. Additional properties. Passed to triggered *MFT-UploadFileEvent*. Supports: • System variables (page 392), e.g. '${PROPERTIES}'. • Properties (page 392), e.g. '$[PROPERTIES]'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Extended settings	True	Use extended settings. Display *URL*, *User name*, and *Password*.
	False	Default. No extended settings.
URL	<URL>	Optional. Editable text field. URL of the (external) WebService. Overwrites the *WebService endpoint*. Supports: • Properties (page 392), e.g. '$[URL]'.

Field name	Value	Description
		• XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). System variables are not supported. Visible when *Extended settings* is switched on.
User name	<UserName>	Optional. Editable text field. User name of HTTP authentication. Properties and system variables are not supported. Visible when *Extended settings* is switched on.
Password	*<Password>*[4]	Optional. Editable password field. Password of HTTP authentication. Properties and system variables are not supported. Visible when *Extended settings* is switched on.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.
3) Semicolon separated key-value list.
4) Secure password.

Preconditions:

- WebServices Client adapter started.

Results:

- Triggers *MFT-UploadFileEvent* on the target system of the WebService endpoint.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Additional properties in case of synchronous upload:

- UPLOADFILE_RETRY_ID := TRACK_ID of current process, part of response from *MFT-UploadFileEvent*.
- UPLOADFILE_RETRY_TRACKID := TRACK_ID of *MFT-UploadFileEvent* process, part of response from *MFT-UploadFileEvent*.

Possible cases of failure:

Possible case of failure	Sync/Async	WebService	Success or error behavior
WebServices Client adapter not started			Error. Throws acUploadFile fault.
User name and password missing	Async	HTTP authentication	Success. Failure when triggering *MFT-UploadFileEvent*.
User name and password missing	Sync	HTTP authentication	Error. Throws acUploadFile fault. Failure when triggering *MFT-UploadFileEvent*.
Unsupported web service configuration	Async	WSS authentication	Success. Failure when triggering *MFT-UploadFileEvent*.
Unsupported web service configuration	Sync	WSS authentication	Error. Throws acUploadFile fault. Failure when triggering *MFT-UploadFileEvent*.
Unsupported web service configuration	Async	Signature/encryption with certificates	Success. Failure when triggering *MFT-UploadFileEvent*.
Unsupported web service configuration	Sync	Signature/encryption with certificates	Error. Throws acUploadFile fault. Failure when triggering *MFT-UploadFileEvent*.

Policies: none.

1.22.4 acUploadFileExt

Caption: Call UploadFileExt endpoint.

Throws Fault=acUploadFileExt.

The *acUploadFileExt* activity is a single file activity that uploads the current process attachment using the specified WebService endpoint. This WebService request triggers an *MFT-UploadFileEvent* (refer to chapter *MFT-UploadFileEvent* in the *MFT Standard Solution - Adapters* manual for more details) on the target system of the WebService endpoint.

Mode of operation:
Performs the upload in synchronous mode only and waits for the response.

Only HTTP authentication is supported. WSS authentication and signing/encrypting with certificates is not supported.

Field name	Value	Description
WebService endpoint	*<Ref>*[1]	Mandatory. Non editable selection field. External WebService endpoints from master data with 'MFTServiceServicePort' port name.
Subject	*<Text>*[2]	Optional. Editable text field. Request subject, description of requested operation. Mapped to *From subject* of triggered *MFT-UploadFileEvent*. Supports: • System variables (page 392), e.g. '${MY_SUBJECT}'. • Properties (page 392), e.g. '$[MY_SUBJECT]'.

Field name	Value	Description
		- XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Properties (key=value;...)	*<KVList>*[3]	Optional. Editable text field. Additional properties. Passed to triggered *MFT-UploadFileEvent*. Supports: - System variables (page 392), e.g. '${PROPERTIES}'. - Properties (page 392), e.g. '$[PROPERTIES]'. - XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Extended settings	True	Use extended settings. Display *URL*, *User name*, and *Password*.
	False	Default. No extended settings.
URL	<URL>	Optional. Editable text field. URL of the (external) WebService. Overwrites the *WebService endpoint*. Supports: - Properties (page 392), e.g. '$[URL]'. - XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). System variables are not supported Visible when *Extended settings* is switched on.
User name		Optional. Editable text field. User name of HTTP authentication. Properties and system variables are not supported. Visible when *Extended settings* is switched on.
Password	*<Password>*[4]	Optional. Editable text field. Password of HTTP authentication. Properties and system variables are not supported. Visible when *Extended settings* is switched on.
Queue	*<Ref>*[1]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Reference to master data.
2) Arbitrary text.
3) Semicolon separated key-value list.
4) Secure password.

Preconditions:

- WebServices Client adapter started.

Results:

- Triggers *MFT-UploadFileEvent* on the target system of the WebService endpoint.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := Text set by *Result* field from acUploadFileExtReply (page 378) activity in triggered *MFT-UploadFileEvent*.
- UPLOADFILE_RETRY_ID := TRACK_ID of current process, part of response from *MFT-UploadFileEvent*.
- UPLOADFILE_RETRY_TRACKID := TRACK_ID of *MFT-UploadFileEvent* process, part of response from *MFT-UploadFileEvent*.

Possible cases of failure:

Possible case of failure	WebService	Success or error behavior
WebServices Client adapter not started		Error. Throws acUploadFileExt fault.
User name and password missing	HTTP authentication	Error. Throws acUploadFileExt fault.
Unsupported WebService configuration	WSS authentication	Error. Throws acUploadFileExt fault.
Unsupported WebService configuration	Signature/encryption with certificates	Error. Throws acUploadFileExt fault.

Policies: none.

1.22.5 acUploadFileExtReply

Caption: Reply to UploadFile call.

Throws Fault=acUploadFileExtReply.

The *acUploadFileExtReply* activity is a single file activity that creates a response for an acUploadFileExt (page 376) activity.

Mode of operation:
The *acUploadFileExtReply* activity has to be used as a part of the *MFT-UploadFileEvent* process (refer to chapter *MFT-UploadFileEvent* in the *MFT Standard Solution - Adapters* manual for more details), i.e. the Criteria *From adapter* field has to set to 'Upload file web service'.

Field name	Value	Description
Result	*<Text>*[1]	Mandatory. Editable text field. Text passed back to acUploadFileExt (page 376) activity of triggering process as $[RESULT] property. System variables, properties and XPath expressions are not supported.
Properties (key=value;...)	*<KVList>*[2]	Optional. Editable text field. Additional properties. Passed back to acUploadFileExt (page 376) activity of triggering process. Supports: • System variables (page 392), e.g. '${PROPERTIES}'. • Properties (page 392), e.g. '$[PROPERTIES]'. • XPath expressions (page 392), e.g. $(upper-case('bookmark')) or $(seeutil:getCurrentDate('yyyyMMdd'))

Field name	Value	Description
		or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')).
Attachment (bookmark)	*<Name>*[3]	Mandatory. Editable selection field. Default value '<current payload>'. Case sensitive name of an existing bookmarked attachment. Passed back to *acUploadFileExt* (page 376) activity of triggering process. Supports: • Properties (page 392), e.g. '$[MY_BOOKMARK_NAME]'. • XPath expressions (page 392), e.g. $(seeutil:getCurrentDate('yyyyMMdd')) or $(seeutil:decimalFormat(seeutil:nextCounter('MYCNT'), '0000000000')). System variables are not supported.
Log message	*<Text>*[1]	Optional. Refer to Common Fields (page 3).

1) Arbitrary text.
2) Semicolon separated key-value list.
3) Arbitrary case-sensitive name except invalid characters ", &, <.

Preconditions:

- The Criteria must match the *MFT-UploadFileEvent* event, i.e. the *From adapter* field has to set to 'Upload file web service'.

Results:

- Sends response to triggering source system.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.
- UPLOADFILE_REPLY_FILE := bookmarked attachment, part of created response.
- UPLOADFILE_REPLY_PARAMETERS := response parameters containing *Result* and *Properties (key=value;...)*.

Possible cases of failure:

Possible case of failure	Success or error behavior
Wrong key-value list syntax in *Properties*	Error. Throws acUploadFileExtReply fault.
Invalid system variable in *Properties*	Error. Throws acUploadFileExtReply fault.
Invalid property in *Properties*	Error. Throws acUploadFileExtReply fault.
Invalid XPath expression in *Properties*	Error. Throws acUploadFileExtReply fault.
Invalid *Attachment(Bookmark)*	Error. Throws acUploadFileExtReply fault.

Policies: none.

1.23 CMA Activities

This section describes activities related to CMA campaigns which, for instance, provide default data and store the collected data. The main use cases are surveys for onboarding campaigns and self-services for data maintenance.

	Survey	**Self Service**
Purpose	Onboarding campaign	Update service
Task	One-time data creation	Repeated updates of user data
Execution	Once	Multiple times
Form initialization	Default data	User data, default data if user data do not yet exist
Data export	Create new data	Create new data and update existing data

Each use case consists of 2 parts: initial population of the CMA form (CMA request type 'INIT') and CMA data export for storing the collected userdata (CMA request type 'EXPORT').

Surveys for Onboarding Campaigns

Initial population needs (in this order):

- Load the default dataset for CMA form population: *acGetBISMasterData* (page 384) activity.
- Convert the dataset into CMA XML format: *acCMAInit* (page 381) activity.

CMA data export needs (in this order):

- Load the default dataset (should be the same as above) to be filled with the CMA request data: *acGetBISMasterData* (page 384) activity.
- Create the CR container XML with the data: *acCMA2ChangeRequest* (page 382) activity with the 'Survey' mode.
- Store the data (CR container XML): *acSetBISMasterData* (page 390) activity.

Self-Services for Data Maintenance

Initial population needs (in this order):

- Load the data for the request sender, if not existing load a default data set: *acGetBISMasterDataForSelfService* (page 385) activity.
- Convert the dataset into CMA XML format: *acCMAInit* (page 381) activity.

CMA data export needs (in this order):

- Load the data for the request sender, if not existing load a template data set (should be the default data set from above): *acGetBISMasterDataForSelfService* (page 385) activity.
- Create the CR container XML with the data: *acCMA2ChangeRequest* (page 382) activity with the 'Self service full CR' or the 'Self service incremental CR' mode.
- Store the data (CR container XML): *acSetBISMasterData* (page 390) activity.

1.23.1 acCMAInit

Caption: CMAInit.

Throws Fault=acCMAInit.

The *acCMAInit* activity is an activity that converts the REST default data XML in the current process attachment into CMA INIT format matching the interaction.xsd from the CMA campaign.

Mode of operation:
Expects the REST data as current process attachment and the CMA INIT request as bookmarked attachment. Creates the CMA XML format and bookmarks the resulting process attachment using "CMA_REPLY" expected by the CMAExternalProcessing web service.

Field name	Value	Description
CMA init request bookmark	*<Name>*[1]	Mandatory. Editable text field. Default value empty. Case sensitive name of an existing bookmarked attachment containing the CMA INIT request XML. Invalid characters ", &, < cannot be entered. Supports: • Properties (page 392), e.g. '$[CMA_BOOKMARK_NAME]'. • System variables (page 392), e.g. '$[CMA_BOOKMARK_NAME]'.
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Arbitrary case-sensitive name except invalid characters ", &, <.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- ScriptExecutor adapter started.
- Current process attachment: REST data XML, default data for CMA form initialization.
- Available bookmark: CMA INIT request.

Result:

- New process attachment: CMA INIT reply.
- CMA_REPLY bookmark: CMA INIT reply. Expected by CMAExternalProcessing webservice.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.
- CMA_REPLY: bookmarked reply XML.

Possible cases of failure:

Possible case of failure	Success or error behavior
Non-existing bookmark name	Error. Throws acCMAInit fault.
Non-existing system variable	Error. Throws acCMAInit fault.

Policy: none.

1.23.2 acCMA2ChangeRequest

Caption: CMA2ChangeRequest.

Throws Fault=acCMA2ChangeRequest.

The *acCMA2ChangeRequest* activity is an activity that converts the CMA export request XML into a change request (CR) XML containing REST data.

Mode of operation:
Extracts the data from the bookmarked CMA export request XML and enters them into the REST template XML contained in the current process attachment. If certificates are to be extracted, too, a certificate section is added to the resulting process attachment. This attachment is a CR wrapper around one or more datasets and can be stored via the *acSetBISMasterData* (page 390) activity (where it is important to use the '/api/mtp' BIS API service).

Field name	Value	Description
Change request mode		Mandatory. Non-editable selection field. Available conversion and data storing types for CMA surveys and CMA self services. All CMA export data are converted into REST data wrapped in a CR container.
	Survey	Creates only new data. Fails when trying to update existing data. To be used for surveys.
	Self service full CR	Creates new data and updates existing data by performing a full merging with the template data or the existing user data. Thus, storing a complete data set is enabled. To be used for self services. Uses a CR container with additional merge=true attribute .

Field name	Value	Description
	Self service incremental CR	Creates new data and updates existing data incrementally i.e. only the new CMA data overwrite the existing REST data fields. To be used for self services. Uses a CR container with additional merge=true attribute.
CMA export bookmark	*<Name>*[1]	Mandatory. Editable text field. Empty by default. Name of a bookmarked attachment containing the CMA export request XML. Supports: • Properties (page 392), e.g. '$[CMA_BOOKMARK_NAME]'. • System variables (page 392), e.g. '${CMA_BOOKMARK_NAME}'.
Embed certificates in CR	True	Additionally convert certificates. Makes *Keystore for embedded certs* visible.
	False	Ignore certificates.
Keystore for embedded certs	*<Keystore>*[2]	Mandatory. Non-editable text field with keystore selection dialog. Target keystore for the exported certificates. Visible when *Embed certificates in CR* is switched on.
Queue	*<Ref>*[3]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Arbitrary case-sensitive name except invalid characters ", &, <.
2) Path to a keystore.
3) Reference to master data.
4) Arbitrary text.

Preconditions:

- Script executor adapter started.
- Current process attachment: REST template XML.
- Available bookmark: CMA export request XML.

Result:

- New process attachment: CR XML containing REST data.
- CMA_REPLY bookmark: attachment above, expected by CMAExternalProcessing web service.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.
- CMA_CR_MODE = value from *Change request target*.
- CMA_REPLY := bookmarked CR XML.

Possible cases of failure:

Possible case of failure	Success or error behavior
Non-existing bookmark name	Error. Throws acCMAChangeRequest fault.
Non-existing system variable	Error. Throws acCMAChangeRequest fault.
Key missing in template for value	Success. Value is ignored.
Key in template misses value in CMA export	Success. Key keeps template default value.

Policy: none.

1.23.3 acGetBISMasterData

Caption: GetBISMasterData.

Throws Fault=acGetBISMasterData.

The *acGetBISMasterData* activity is an activity that loads the specified REST dataset from BIS as an XML attachment.

Field name	Value	Description
Template dataset ID	*<DatasetID>*[1]	Mandatory. Editable text field. ID for REST dataset to be loaded. Default dataset matching the needs of the calling CMA survey. Format for AS2 data: /api/as2/<own-AS2ID>/<Partner-AS2ID>. Supports: • Properties (page 392), e.g. '$[AS2_DATASET_ID]'. System variables are not supported.
Queue	*<Ref>*[2]	Mandatory. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid REST dataset ID.
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- HTTP Client adapter started.
- System variable ${CMA_REST_HTTP_CONNECTION}: Master data HTTP connection for REST data loading.

- System variables CMA_REST_HTTP_USER and CMA_REST_HTTP_PWD containing REST service authentication.

Result:

- New process attachment: loaded REST data XML.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Erroneous HTTP connection	Error. Throws acGetBISMasterData fault.
Wrong user or password	Error. Throws acGetBISMasterData fault.
Invalid dataset ID	Error. Throws acGetBISMasterData fault.

Policy: none.

1.23.4 acGetBISMasterDataForSelfService

Caption: GetBISMasterData ForSelfService.

Throws Fault=acGetBISMasterDataForSelfService.

The *acGetBISMasterDataForSelfService* activity is an activity that loads the REST dataset belonging to the CMA request sender as an XML attachment.

Mode of operation:
Extracts necessary data from the CMA request XML expected as current process attachment. Checks whether the REST data for the CMA request sender do already exist. If they do, these REST data are loaded. If they do not, the default data specified by the *Default dataset ID* are loaded.
The reference from the CMA request sender to the REST data is stored in a special SharePoint entity hierarchy. Refer to **SharePoint Entity Hierarchy Reference** below for more details.

Field name	Value	Description
TPM share point	*<Name>*[1]	Mandatory. Non-editable selection field. Available CM_SHARE_POINT entities and an auto-create mode entry. The entities are the root entities of the TPM reference hierarchy and need to be created manually using the CM-SHARE-POINT entity class.

Field name	Value	Description
	<use default>	Default. Uses the default TPM sharepoint root node which is automatically created if does not already exist.
Default dataset ID	*<DatasetID>*[2]	Mandatory. Editable text field. ID for the default REST dataset to be loaded when the data for the CMA request sender are not existing. Note that the default dataset must match the needs of the calling CMA self-service. Format for AS2 data: /api/as2/<own-AS2ID>/<Partner-AS2ID>. Supports: • Properties (page 392), e.g. '$[AS2_DATASET_ID]'. System variables are not supported.
Queue	*<Ref>*[3]	Mandatory. HTTP queue for getting REST master data. Refer to Common Fields (page 1). I It is recommended to use an own queue that is not used for other purposes and which does not retry a failed order.
Abort on error	True/False	Refer to Common Fields (page 3). ⚠ Switching off this field can lead easily to overwriting AS2 data of other CMA users. Be very careful when designing such processes.
Log message	*<Text>*[4]	Optional. Refer to Common Fields (page 3).

1) Arbitrary case-sensitive name except invalid characters ", &, <.
2) Valid REST dataset ID.
3) Reference to master data.
4) Arbitrary text.

Preconditions:

- HTTP Client adapter started.
- System variable ${CMA_REST_HTTP_CONNECTION}: Master data HTTP connection for REST data loading.
- System variables ${CMA_REST_HTTP_USER} and ${CMA_REST_HTTP_PWD}: REST service authentication.

Result:

- New process attachment: loaded REST data XML.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.
- CMA_RESULT := error information if creating the TPM failed.
- CMA_DEFAULT_DATA_USED := 'TRUE' if the default REST dataset has been loaded.
- CMA_DEFAULT_DATA_USED := 'FALSE' if existing user data have been loaded.

Properties from the CMA request XML:

- CMA_FORM_NAME := REST form name (CMA form data).
- CMA_ORGANIZATION := Name of user organization (CMA user data).
- CMA_PROJECT_NAME := Name of CMA project (CMA user data).
- CMA_PROJECT_ID := ID of CMA project (CMA user data).
- CMA_USER_EMAIL := User email (CMA user data).
- CMA_USER_REFID := User refId (CMA user data).
- CMA_OWN_AS2_ID := Own AS2 ID (CMA form data).
- CMA_PARTNER_AS2_ID := Partner AS2 ID (CMA form data).

Possible cases of failure:

Possible case of failure	Success or error behavior
Erroneous HTTP connection	Error. Throws acGetBISMasterDataForSelfService fault.
Wrong user or password	Error. Throws acGetBISMasterDataForSelfService fault.
Invalid dataset ID	Error. Throws acGetBISMasterDataForSelfService fault.

Policy: none.

SharePoint Entity Hierarchy Reference

The TPM entity hierarchy reference to the REST user data consists of 4 levels:

Level	Entity Class	Label	Description	Content
1	CM-SHARE-POINT	<RootName>[1]	Root node. There is a default root node that can be used. Either, root nodes can be created manually. Existing root nodes are selectable in *TPM share point*.	cmContact[2]
2	CM-ORGANISATION	Organization:<orgName>	Contains the organization name from the CMA request XML. Created and selected automatically.	cmContact[2]
3	CM-ORGANISATION-UNIT	Project: <CMAProjectName>	Contains the project name from the CMA request XML. Created and selected automatically.	cmContact[2]
4	CM-SHARED-DATA	Form: <FormName>	Contains the form name from the CMA request XML. Created and selected automatically.	cmDataReference[3]

1) In case of *TPM share point* set to "<use default>" the label is "CMAExport".
2) Contact field with name, address, email, phone, fax, organization and project filled with data from the CMA request XML.
3) cmDataReference field with project, identifier and dataset. Identifier is filled with the form name. In case of AS2, the last label is set to 'AS2 data set'. The value is the AS2 identifier and denotes the reference to the AS2 REST data.

TPM and AS2 data creation and modification

TPM	AS2 record	Success or error behavior
Not existing	Not existing	Success. TPM and AS2 record are created.
Not existing	Existing	Error. Throws acGetBISMasterDataForSelfService fault.
Existing	Not existing	Success. AS2 record is created.
Existing, AS2 identifier not changed	Existing	Success. AS2 record is updated.
Existing, AS2 identifier changed	Does not matter	Error. Throws acGetBISMasterDataForSelfService fault.

⚠ It is not recommended to allow that 2 data references have the same AS2 identifier. If you switch off the *Abort on error* be very careful to not allow overwriting other AS2 data.

1.23.5 acUpdateSelfServiceData

Caption: UpdateSelfServiceData.

Throws Fault=acUpdateSelfServiceData.

⚠ DEPRECATED! Do not use!

The *acUpdateSelfServiceData* activity is an activity that stores the reference from the CMA request sender to the stored REST data in a special entity hierarchy.

Field name	Value	Description
Dataset ID	*<RestID>*[1]	Mandatory. Editable text field. Default value empty. The basic path of a REST dataset (e.g. '/api/as2' in case of AS2 data). Supports: • Properties (page 392), e.g. '$[AS2_DATASET_ID]'. System variables are not supported.
Request bookmark	*<BookmarkName>*[2]	Optional. Editable text field. Empty by default. Name of a bookmarked attachment containing the CMA request XML. Supports: • Properties (page 392), e.g. '$[BOOKMARK]'. • System variables (page 392), e.g. '${BOOKMARK}'.
Log message	*<Text>*[2]	Optional. Refer to Common Fields (page 3).

1) Basic path for REST dataset, format: '/api/<comp>'.
2) Arbitrary text.

Preconditions:

- HTTP Client adapter started.
- Bookmarked attachment: CMA request XML.
- Current process attachment: CMA request XML. Ignored when a bookmark is specified.

Result:

- Process attachment: not changed.

Sets the following properties:

- STATUS := 'SUCCESS'.
- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Erroneous dataset ID	Error. Throws acUpdateSelfServiceData fault.
Non-existing property	Error. Throws acUpdateSelfServiceData fault.
Non-existing system variable	Error. Throws acUpdateSelfServiceData fault.

Policy: none.

Generic Data Reference

The reference from the CMA request sender to the data to be updated is stored in a special entity hierarchy. The data are extracted from the CMA request XMl in either the bookmarked attachment or the current process attachment.

Root node	Name in case of AS2: 'SharePointAS2'	CM-SHARE-POINT	
Level 1 node	Name: 'User:<UserRefID>'	CM-ORGANISATION	Contact with: • User data from CMA request • Project name in Org. unit field
Level 2 node	Name: <CMAProjectName>	CM-ORGANISATION-UNIT	Contact with: • User data from CMA request • Project name in Org. unit field
Level 3 node: Data node	Name: <FormName>	CM-SHARED-DATA	cmDataReference with: • Label: CMA project name • Identifier: name of REST form XML file • AS2 data set: AS2 identifier

Example for a user dataset path: '/SharePointAS2/User:d8a56240-0596-11e7-b328-ab870a006873/AS2 Self Service WS/ frmAS2RestPartner.xml'.

1.23.6 acSetBISMasterData

Caption: CMASetBISMasterData.

Throws Fault=acSetBISMasterData.

The *acSetBISMasterData* activity is an activity that stores the REST data in the current project attachment directly or as change request depending on the specified *BIS API service*.

Field name	Value	Description
BIS API service	*<APIService>*[1]	Mandatory. Editable text field. Default value '/api/mtp'. The basic path of a REST dataset ID (e.g. '/api/as2' or '/api/mtp') representing the storing target. In case of change requests the MTP ID has to be used: '/api/mtp'. This is recommended for CMA survey campaigns and self-service projects especially when, for instance, storing AS2 data with certificates. If the data have been generated by the *acCMA2ChangeRequest* (page 382) activity this is absolutely necessary because a CR container is wrapping the actual data. Supports: • Properties (page 392), e.g. '$[MTP_SERVICE]'. System variables are not supported.
Queue	*<Ref>*[2]	Mandatory. HTTP queue for storing master data. Refer to Common Fields (page 1).
Log message	*<Text>*[3]	Optional. Refer to Common Fields (page 3).

1) Valid path to BIS API service (generally matching '/api/*').
2) Reference to master data.
3) Arbitrary text.

Preconditions:

- HTTP Client adapter started.
- Current process attachment: REST data XML.
- System variable ${CMA_REST_HTTP_CONNECTION}: Master data HTTP connection for REST data storing.
- System variables ${CMA_REST_HTTP_USER} and ${CMA_REST_HTTP_PWD}: REST service authentication.
- Property $[CMA_CR_MODE]: set by *acCMA2ChangeRequest* (page 382) activity (affects use of PUT/POST method of HTTP request).

Result:

- Process attachment: not changed.

Sets the following properties:

- MD_STATUS_CODE := Status code from storing.
- MD_STATUS_TEXT := Status text from storing.
- STATUS := 'SUCCESS'.

- RESULT := '100'.

Possible cases of failure:

Possible case of failure	Success or error behavior
Erroneous HTTP connection	Error. Throws acSetBISMasterData fault.
Wrong user or password	Error. Throws acSetBISMasterData fault.
Invalid Dataset ID	Error. Throws acSetBISMasterData fault.
Invalid value in REST data	Error. Throws acSetBISMasterData fault.

Policy: none.

1.24 Appendix

TODO

1.24.1 Properties, System Variables and XPath Expressions

1.24.1.1 Notation

${} – References a system variable. System variables are global. They are specified in the GUI for all processes.

$[] – References a property. Properties are local, their values are available for the currently running process instance only. They are located in a special process variable called property bag. Only properties added to such a property bag are accessible in the process. In the MFT solution, all activities have a property bag.
New properties can be added to the property bag, for example with a mapping. All output values specified in the mapping are returned to the process and added to its property bug. Thus, the properties in the property bag are accessible for all subsequent activities.

$() – Specifies an XPath expression. For instance, COUNTER is defined as a system variable which expands to an expression $(seeutil:decimalFormat(seeutil:nextCounter('COUNTER'), '0000000000')). An overview of available XPath functions can be found in the SEEBURGER XPath Functions book (link).

1.24.1.2 System Variables

1.24.1.2.1 General MFT System Variables

System Variable	Default Value	Description
MFT-PROCESSOR-DESTINATION	direct:	Because logging is made via a mapping this is the queue
MFT-REGISTER-TRIGGER-DIRECT	TRUE	Currently only TRUE is supported
MFT-TPM-ONLY-ONE-MATCH-ALLOWED	FALSE	If set to TRUE lookup CriteriaMFT will fail if more than one entity matches the criterion.
MFT-TRACKID-COUNTER	$(concat(seeutil:getEnv(), seeutil:decimalFormat(seeutil:nextCounter ('MFT_TRACKING_COUNTER'), '0000000')))	The XPath expression which creates the process TRACKID.

1.24.1.2.2 Message Tracking Related System Variables

System Variable	Default Value	Description
SSC-MT-ROOT-PATH	../../../../MT_ROOT/	Message tracking root folder – this is the correct path if message tracking is installed together with BIS6 below the same root folder.
MFT-LOG-SUB-PATH	/mft.log/	Sub-path for log files below SSC-MT-ROOT-PATH
MFT-MT-DATA-SUB-PATH		A predefined sub path below SSC-MT-ROOT-PATH for storing payload data. The *acTracker* (page 47) activity uses this path.
MFT-MTLOGGER-DESTINATION	direct:	sync: or direct: defines the bic queue for message tracking
MFT-MTLOGGER-MAPPING	CC_MT_Mapping	Mapping name used for message tracking
MFT-MT-LOG-NOTIFICATIONS	TRUE	If TRUE, a MT entry is written in case the component replies with status SUCCESS_NOTIFY (see activity development framework)

1.24.1.2.3 Activity Specific System Variables

System Variable	Default Value	Description
MFT-FILELISTCREATOR	MFT-FILELISTCREATOR	The mapping which is used to create the standard file list used in *scForEach* (page 10).
MFT-AS2-MDN-HANDLING	AFTER_RECEIVE, GATEWAY_MODE	In case AS2 sender requested an asynchronous MDN: AFTER_RECEIVE – the MDN is sent immediately after process started (always positive MDN). GATEWAY_MODE – the MDN is sent after the process is finished. (in case process failed and a fault handling was configured to resumes the process a negative MDN can be sent).
MFT-EUN-EXTERNAL-PROGRAM-SCRIPT	CallProgram.bsh	The beanshell script which is used in acRunExternalProgramm component

1.24.1.3 Process Properties

These properties are created from the initiating event and remain the same during process execution.

- Part 1: File related adapters (hotfolder, file system, FTP, SFTP)
- Part 2: Several DT adapters (AS2, AS4, JMS, MAIL, OFTP, X400)
- Part 3: HTTP and SAP adapters
- Part 4: Various remaining adapter types (SLMP, SEEFX, WebService, Time Event, Database Event)

1.24.1.3.1 File Related Adapters

All mainly file related adapters with similar properties.

	Hotfolder (classic)	**FileSystem (Server and client)**	**FTP (Client)**	**FTP (Server)**	**SFTP (Client)**	**SFTP (Server)**
FROM_ADAPTER	HOTFLD	FILSYS	FTPCLI	FTPSRV	SFTCLI	SFTSVR
FROM_NAME	Name of folder listener i.e. groupname of hofolder event data	BIS address name of file request or 'NOT_SET'	BIS origin address name or 'NOT_SET'	BIS origin address name	BIS origin address name	BIS origin address name or 'NOT_SET'
FROM_UUID	UUID of folder listener i.e. groupID of hotfolder eventData	BIS address UUID of file request or 'NOT_SET'	BIS origin address UUID or UUID placeholder[1]	BIS origin address UUID	BIS origin address UUID or UUID placeholder[1]	BIS origin address UUID
FROM_ID	Same as FROM_NAME	unused	User name of FTP request partnerSettings (configured in FTP address)	User name of FTP request partnerSettings (configured in FTP address)	User name of SFTP request partnerSettings (configured in SFTP address)	User name of SFTP request partnerSettings (configured in SFTP address)
FROM_PATH	File name of hotfolder eventData	File name of file request file parameter	Remote file name of FTP request without path	File name of FTP request	Remote file name of SFTP request	File name of SFTP request
FROM_SUBJECT	File path without file name ending with /	Full path of file request file parameter, ending with a path separator '/'	Sub-path of user home folder ending with / ex. /USER1/ (may be empty)	Sub-path of user home folder ending with / ex. /USER1/ (may be empty)	Sub-path of user home folder ending with / ex. /USER1/ (may be empty)	Sub-path of user home folder ending with / ex. /USER1/ (may be empty)
TO_NAME	hotfolderName of hotfolder eventData	unused	unused	unused	unused	unused
TO_UUID	hotfolderID of hotfolder eventData	unused	unused	unused	unused	unused
TO_ID	unused	unused	unused	unused	unused	unused

1) UUID placeholder '????????-????- ????-????- ????????????'.

1.24.1.3.2 DT adapters

Several DT adapters except HTTP, FTP and SFTP.

	AS2 (Server)	AS4 (Server)	JMS	MAIL (Client)	MAIL (Single Flow)	OFTP TCP/IP	X400 / P7
FROM_ADAPTER	AS2SRV	AS4SRV	JMSCLI	MAILCL	MAILSF	OFTSRV	X400P7
FROM_NAME	AS2 request: origin address name	AS4 message senderInformation: address name	JMS message: destination address name	MAIL message: origin address name or 'NOT_SET'	MAIL message: origin address name or 'NOT_SET'	OFTP request: origin address name	P7 request: origin address name
FROM_UUID	AS2 request: origin address ID	AS2 message senderInformation: address ID	JMS message: destination address ID	MAIL message: origin address ID or UUID placeholder[1]	MAIL message: origin address ID	OFTP request: origin address ID	P7 request: origin address ID
FROM_ID	AS2 request: originator AS2 ID	AS4 message senderInformation: party ID	*Identical to FROM_NAME*	MAIL message: original address email	MAIL message: original address email	OFT request: sfidCode (originator) or 'NOT_SET'	P7 request originatorAddress: surName or 'NOT_SET', '/' and givenName appended if givenName is available
FROM_PATH	AS2 request: file name or 'NOT_SET'	'NOT_SET'	JMS message: provider queue	MAIL message: payload name or empty	MAIL message: payload name or 'NOT_SET'	OFTP request: dataSetName (transferInfo) or 'NOT_SET'	P7 request omsCdifInformation: fileName or comment, fileName/ comment if available, 'NOT_SET' if not available
FROM_SUBJECT	AS2 request subject	AS4 message: palyoad file name or 'NOT_SET'	Specified by system variable[2] (see examples below[3])	MAIL message: subject	Mail message: subject	OFTP request: subject or 'NOT_SET'	P7 request: subject or 'NOT_SET'
TO_NAME	BIS destination address name	AS4 receiver address name	#	BIS destination address name	Destination address name	BIS destinator address name	BIS own address name
TO_UUID	BIS destination address UUID	AS4 receiver UUID	unused	BIS destination address UUID	Destination address UUID	BIS destinator address UUID	BIS own address UUID
TO_ID	AS2 ID from AS2 request recipient	AS4 receiver party ID	#	Destination email address: mailbox used from received mail message mailboxOwner settings	Mailbox user	SFID code from OFTP request destinator	P7 request: destination address, 'sub name/ given name' if available, 'NOT_SET' if none is available

1) UUID placeholder '????????-????- ????-????- ????????????'.
2) System variable ${MFT-SUBJECT-PATTERN-JMS}.
3) Examples: "$[GROUP_ID]" or "$[MESSAGE_ID]" or "${JMS_REPLY_ DESTINATION}"

1.24.1.3.3 HTTP and SAP adapters

HTTP and SAP client adapters as well as synchronous and asynchronous server adapters.

	HTTP (Client)	**SHC-Sync. (HTTP Server)**	**SHC-Async. (HTTP Server)**	**IDOC**	**SAP RFC (Async.)**	**SAP RFX (Sync.)**
FROM_ADAPTER	HTTPCL	SHCSYN	SHCASY	IDOCTL	RFCCTL	RFCSYN
FROM_NAME	HTTP request: destination address name	SHC request: destination address name	SHC request: destination address name	IDOC event: registration name	RFC request: SAP client or 'NOT_SET'	RFC request: SAP client or 'NOT_SET'
FROM_UUID	HTTP request: destination address UUID	SHC request: destination address UUID	SHC request: destination address UUID	IDOC event: registration ID	RFC request: address ID	RFC request: address ID
FROM_ID	HTTP request: user (clientInfo) or 'NOT_SET'	SHC request: remote host (serverInfo)	SHC request: remote host (serverInfo)	*Identical to FROM_NAME*	RFC request: connection ID	RFC request: connection ID
FROM_PATH	'NOT_SET'	SHC request: requestURI or 'NOT_SET'	SHC request: requestURI	IDOC event: program ID	RFC request: function name or 'NOT_SET'	RFC request: function name or 'NOT_SET'
FROM_SUBJECT	HTTP request: subject or 'NOT_SET'	SHC request: queryString or 'NOT_SET'	SHC request: queryString or 'NOT_SET'	Specified by system variable[1], semicolon separated list (see example below[2])	RFC request: program ID or 'NOT_SET'	RFC request: program ID or 'NOT_SET'
TO_NAME	unused	unused	unused	'NOT_SET'	unused	unused
TO_UUID	unused	unused	unused	'NOT_SET'	unused	unused
TO_ID	unused	unused	unused	'NOT_SET'	unused	unused

1) System variable ${MFT_SUBJECT_ PATTERN_IDOC}.
2) Example for semicolon separated list: 'MESCOD=$[SAP_MESCOD];MESFCT=$[SAP_MESFFCT];...". The properties inside are resolved by extracted IDOC metadata.

1.24.1.3.4 Various adapter types

Collection of the remaining adapter types:

	SLMP SEE LINK	SEE FX	BIS FX	Upload File WebServic	Generic Web Service	Splitter	Time Event (Scheduler	Database Event
FROM_ADAPTER	SLMPCL	SEE-FX	BIS-FX	UPLOAD	WEBSRV	SPLFIL	TIMEVT	DBEVNT
FROM_NAME	SLMP message: address name (originator)	Name of creator of file (createdBy of download file response FileInfo)	SHC request: destination address name or empty	fromName (see wsdl)	Decider rule name	fromName	PARAM1 (in scheduler config)	JDBC request DBListenerEvent: port name
FROM_UUID	SLMP message: address ID (originator)	Not set	SHC request: destination address id	UUID placeholder[1]	UUID placeholder[1]	fromUUID	unused	JDBC request DBListenerEvent: port ID or empty
FROM_ID	SLMPmessage: SLMPID (originator)	Not set	SHC request: query string for 'operator' or 'NOT_SET'	Id (see wsdl)	Port type	fromID	unused	JDBCD request DBListenerEvent: engine ID or 'NOT_SET'
FROM_PATH	SLMP message: payload file name or 'NOT_SET'	File name (from download file response FileInfo)	SHC request: query string for 'path' or 'NOT_SET'	Path (see wsdl)	Address (EndPoint Reference)	fromPath	PARAM2 (in scheduler config)	JDBC request DBListenerEvent: data source name or 'NOT_SET'
FROM_SUBJECT	SLMP message: subject	SEE FX path without file name ending with /	SHC request: query string for 'operation' or 'ReceiveUpload'	Subject (see wsdl)	Operation	fromSubject	Name of scheduler task (in scheduler config)	JDBC request DBListenerEvent: table name or 'NOT_SET'
TO_NAME	BIS address name from SLMP request recipient	folder of event or 'My Files'	SHC request: query string for 'sysuser' or empty	Can be set via property TO_NAME (see wsdl)	unused	unused	PARAM3 (in scheduler config)	'NOT_SET'
TO_UUID	BIS address UUID from SLMP request recipient	unused	'NOT_SET'	unused	unused	unused	unused	'NOT_SET'
TO_ID	SLMPID from SLMP request recipient	unused	SHC request: query string for 'e-mail' or empty	Can be set via property TO_ID (see wsdl)	unused	unused	unused	'NOT_SET'

1) UUID placeholder 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'.

1.24.2 SEE FX Path Guidelines

This section gives an overview of SEE FX paths including virtual paths.

Paths

A path, as used within this API, uses a single slash (“/”) as path separator. All paths are absolute and thus must start with a leading “/”. The root path (“/”) is the user directory. At the time of writing this document its content are the special folders:

- My Files
- My Subscriptions
- My Shares
- My Attachments
- Recycle Bin
- Upload [1]
- Downloads [1]

This list might change in the future, so please refer to the documentation of your BIS FileExchange version for more information about which folder will be present.

Paths to directories end with “/”, paths to files do not end with “/”.

As paths are required to be unique, a special notation is used for folders in *My Subscriptions*. The name of a subscription is the folder name followed by '#' followed by the owner user ID, e.g. "*/My Subscriptions/Test#root/aFile.txt*".

Examples:

- Path to Folder: /My Files/BIS/
- Path to File: /My Files/test.txt

Note: Some special paths require certain authorities in order to be accessed. Please find detailed description of available BIS FileExchange Authorities in the Identity Management Manual or contact your BIS FileExchange administrator if you experience authority-related problems with the BIS FileExchangeX Web Services.

[1] Special folders /Upload and /Downloads are described in detail in the next sub section.

1.24.3 Virtual Paths in SEE FX Web Service API

Warning: This feature is in status experimental. It was added so that clients operating behind less-powerful protocols (e.g FTP) can send and receive files to/from other SEE FX users (either regular or technical). It is tested to work with the FileTransferService3 web service. No virtual paths are yet supported in the other SEE FX Web Services.

Note: For a general description of paths, used within the SEE FX Web Service API, please refer to topic General:Paths

Virtual paths are paths that do not represent an existing SEE FX file or folder. Instead, they are either special-purpose folders, or they point to an existing SEE FX file, from a different path - compatible to the purpose.

Each supported virtual path is also protected by access restrictions (e.g some are read-only, others are write-only etc.).

Note: By assumption, virtual folders are not deletable themselves, as they do not represent a real, existing SEE FX folder, that can be deleted.

Please find below a list of all currently supported virtual paths, their access restrictions and brief description of the purpose of each.

Path	Description	Read	Write	Delete	Permit	View	Required Authority
/Downloads/	Virtual inbox of the caller. Contains all files, sent to the caller by other SEE FX users.	yes	-	yes	-	-	SEEFX_RECEIVEFILE
/Downloads/Read Files/	Contains files sent to the caller by other SEE FX users, that have already been read - downloaded, deleted, copied.	yes	-	yes	-	-	SEEFX_RECEIVEFILE
/Downloads/Unread Files/	Contains files sent to the caller by other SEE FX users, that have been delivered to your inbox, but not yet processed.	yes	-	yes	-	-	SEEFX_RECEIVEFILE
/Upload/	Parent to the "send-folder" hierarchy. Read-only.	yes	-	-	-	-	SEEFX_SENDFILE
/Upload/Users/	Parent to the virtual inboxes of all SEE FX users, that the caller is allowed to send to. Read-only.	yes	-	-	-	-	SEEFX_SENDFILE

Path	Description	Read	Write	Delete	Permit	View	Required Authority
/Upload/Users/<username>/	Virtual inbox of the SEE FX user, whose username matches the name of this folder. Write-only. Files uploaded here are direclty sent to the user.	-	yes	-	-	-	SEEFX_SENDFILE
/Upload/Groups/	Parent to the virtual inboxes of all SEE FX groups, that the caller is allowed to send to. Read-only.	yes	-	-	-	-	SEEFX_SENDFILE
/Upload/Groups/<group name>/	Virtual inbox of the SEE FX group, whose name matches the name of this folder. Write-only. Files uploaded here are direclty sent to all SEE FX users, that are members of this group.	-	yes	-	-	-	SEEFX_SENDFILE

1.24.4 In-Sequence Processing

1.24.4.1 Introduction to Event Sequencing and In-Sequence Processing

An event is a request or a response sent from one component to another. Event sequencing means that the target component has to process events in the same order in which they have been sent by the source component. These events may not be processed out of order (e.g. subsequent master data updates). Otherwise, errors or exceptions will occur (race conditions, e.g. leading to erroneous or spoiled data).

Event sequencing is accomplished by creating sequences. A sequence is characterized by a name (sequence name) and a monotonically ascending and gap-free number (sequence number), generally based on a counter. Sequence configuration is done by so-called sequence profiles.

In general a process which wants make use of the In-Sequence component has to be designed for it (Process Designer).

A process makes use of a sequence by passing the name of the sequence together with a sequence number. The component then decides if this process has to be interrupted or may continue. If the process is allowed to continue the sequence gets locked for all other processes which are calling the In-Sequence component with the same sequence name. The process which continued has to call the In-Sequence component again in order to free the sequence with this name. Then the In-Sequence component resumes the process which sequence number is next without gap. If this process did not yet arrive no other process can pass the In-Sequence component.

Therefor it must be possible to handle several cases which can occur. This can be configured in the profile assigned to the sequence.

Such cases are:

- The expected sequence number does not arrive
- Sequence numbers of the past (< current sequence number) arrive.
- Sequence numbers much higher than the current sequence number arrive
- A process holding the lock for a sequence does not release the lock in time.

How such disorders of sequences happen, may have various reasons and depending on the reasons different ways how to react.The BIS in Sequence processing offers therefore multiple options to configure the behavior of a sequence if such events occur.

In all cases alerts are triggered which can be caught via BIS alerting and these alerts also appear in the BIS GUI dashboard.

Handling of those events can be automatic (by configuration of the profile) or manually. All manual action can be performed via BIS-GUI in

Monitoring | Sequence Processing | Sequence state.

This includes:

- Skipping a gap via the BIS Front-end.
- Releasing a blocked sequence by terminating the process which holds the lock.

Besides this it is possible to manually pause and resume a certain sequence.

1.24.4.1.1 Overview

Main components:

- Configuration: Sequence profiles configurable in the BIS GUI > Configuration tab > Sequence processing
- Monitoring: Sequence state monitoring screen in the BIS GUI > Monitoring > Sequence processing

Configuration and monitoring are visible/editable in case of the following user rights 'SHOW_SEQUENCE' and 'SEQUENCE_EDIT_ALLOWED'.

Main use cases:

- FIFO processing for received files:

An adapter receives files/messages which are to be processed in the order of their reception. When a file/message is received the adapter generates a new sequence number and assigns it to the file/message.

- Sending component determines the order of processing:
  Received files contain the sequence number that defines the order of processing them. The sequence number is extracted from the received file using a mapping or a content extractor.
- A variant of the case above: Protocol metadata of the received message contain the sequence number.
- Data need to be sent and applied in an exactly determined order that may not be changed, for instance master data updates.

1.24.4.1.2 Sequence Profiles

Sequence configuration GUI located on the Configuration tab > Sequence processing > Sequence profiles. Allows configuration of the following items:

- Critical section timeout handling
- Sequence gap handling
- Out-of-sequence handling
- Instance-down handling
- Reorganization of process sequences.

The following fields are provided (the administration tab fields have been omitted, they are not relevant).

Name

Value	Description
<Text>	Mandatory. Editable text field. Name of the sequence profile.

Timeout for leaving the critical section

Value	Description
<Int>[2]	Editable number field. Default value 60 seconds. Minimum value 10 seconds. Maximum value 999999 seconds (277,77 hours or 11,57 days). Timeout for critical section/sequence scope processing in seconds. Actually the maximally allowed runtime for critical section processing.

Timeout handling

Non-editable selection field. Critical section timeout handling. Provides several options for action in case a critical section/sequence scope timeout occurs.

Value	Description
Keep lock and abort process with error	Default. The process is aborted with error, and its state is set to 'Stopped with error (9)'. The state of the current sequence element that caused this timeout is set to 'Timed out'. The critical section lock is kept, and the expected sequence number is not changed. No other process can enter the critical section. Manual user interaction is needed, for instance, the process can be continued after fixing the error.
Release lock and abort process with error	The process is aborted with error, and its state is set to 'Stopped with error (9)'. The state of the current sequence element that caused this timeout is set to 'Timed out'. The critical section lock is released, and the expected sequence number is increased by 1. The critical section is free. The next (waiting) process is allowed enter the critical section. If the process of the timed out sequence element is continued an OOS condition is produced. Further processing depends on OOS configuration (see OOS fields below).
Send notification, reset timeout	An alert is raised. The timeout timer is restarted. The currently processed sequence element is not aborted. When the timeout is reached, another alert is raised, and the timeout timer is restarted. This timeout rotation is only interrupted if the current sequence element finishes processing, or if it is manually aborted.

Max allowed gap

Value	Description
<Int>[2]	Editable number field. Default value 1. Minimum value 1. Maximum value 999999. Maximum size of sequence number gap that is allowed to be handled successfully. Greater gaps cause processes to be stopped with error. The sequence element is set to 'Failed' state.

Timeout for resolving gap

Value	Description
<Int>[2]	Editable number field. Default value 600 seconds. Minimum value 10 seconds. Maximum value 999'999 seconds (277.77 hours or 11.57 days). Timeout for resolving sequence number gaps in seconds. If a timeout occurs, the process is stopped with error, the process state is set to 'Stopped with error (9)', and the sequence element state is set to 'TimedOut'.

Timeout handling for gap

Non-editable selection field. Provides several options of action in case a sequence number gap timeout occurs.

Value	Description
Skip all missing elements	Default. Start the timeout once for the whole gap. Skip all still missing elements after the timeout occured. The currently waiting sequence element (with 'Pending' state) that caused the gap may enter the critical section.
Skip next element	Skip the next element after the timeout occured. Start a new timeout countdown for the next missing element. If no missing element arrives, the total waiting time for the waiting sequence element that caused this timeout is: MissingElementCount * Timeout seconds. Be careful when using this timeout switch: if the sequence profile allows large gaps this can result in a very long runtime.
Send notification, reset timeout	An alert is raised. The timeout timer is started, and the sequence waits for the missing element. If the timeout occurs again, another alarm is raised, and the timeout timer is restarted again. This timeout rotation is only interrupted if either the expected sequence element arrives (it is allowed to enter the critical section and is processed) or it is manually skipped. Only after an interruption, normal sequence processing is resumed, and the waiting sequence element that caused the gap is allowed to enter the critical section and being processed.

Handling of out-of-sequence elements

Non-editable selection field. Provides several options of action in case of processes with sequence numbers that are out of sequence, i.e. curSeqNo < expSeqNo, but the sequence numbers are no duplicates of already existing sequence elements.

Only sequence number comparation takes place. This is no duplicate checking, the content is not compared at all.

Value	Description
Abort process with error	Default. The element is rejected by the sequence with an appropriate error message. The process is stopped with error. Process status is 'Stopped with error (9)'. The sequence elements are set to 'Failed' state.
Process as next	The process may enter the critical section and is processed in order to fill the gap. The current/ expected sequence number is not changed.

Handling if instance is down

Non-editable selection field. Proves several options for action in case an active instance is shut down.

Value	Description
Skip current element	Default. Skip the element if its assigned Process-Engine is shut down.
Wait until the instance is back	Do nothing and wait until the assigned Process-Engine is active again.

Reorganize sequence

Non-editable selection field. Provides several reorganization options.

Value	Description
Do not reorganize	Default. No reorganization of sequences and sequence element.
Reorganize after time	Reorganize sequences and sequence elements after a configurable amount of time. Makes *Reorganize time* visible.

Reorganize time

Value	Description
<*Int*>[2]	Editable number field. Default value 60 minutes. Minimum value 0. Maximum value 999999. The amount of time after which sequences and sequence elements are reorganized. Visible when *Reorganize sequence* is set to 'Reorganize after time'.

2) Positive Integer number.

1.24.4.2 Modelling a Process Sequence

Modelling a process sequence by configuration of sequence profiles. Covers several general use cases as well as pitfalls.

When modelling desired sequencing behavior the following items are to be considered:

1. How many sequence elements are expected to arrive in which time?
   The resulting numbers affect sequence gap and OOS handling: What should the preferred gap size be? Should processing OOS elements be allowed? Should it be possible to fill the gaps?
2. What is the arbitrary and what is the greatest expected runtime of the sequence scope?
   The resulting numbers affect the critical section timeout: What is the timeout?

Pitfalls:

1. Many sequence elements arrive in a short time. Most likely, they do not enter the critical section in ascending order of the sequence number. This can cause:

- Gaps, that have to be filled => the sequence profile must allow gap filling
- Sequence elements are rejected because their sequence number is too great, i.e. above the allowed gap size => Max gap must be great enough to minimize rejected sequence elements.

- There are use cases where gaps are not allowed and sequence processing in a not strictly ascending order of sequence numbers lead to spoiled data, for instance, master data updates => Gaps must be avoided. Each sequence element is only allowed to enter the critical section if all its predecessors have been processed (except the very first sequence element, of course). Set the gap resolving timeout to a rather high value. This allows the missing sequence elements to arrive in time.

2. Sequence elements with sequence numbers that do not fit in (outlier elements, much too great or much too small):

- Carefully determine the restrictions of gaps in case of sequence numbers that are too great (MaxGap).
- Sequence numbers that are much too small: 2 OOS options are available. Either all OOS elements are treated as errors, or all OOS elements are processed as next element. Thus, these outlier elements can not be distinguished from the other OOS elements.
  If it is desired to reset the sequence / sequence number, create a new sequence by changing the sequence name. Sequence numbers will restart from the very beginning, called the 'golden transaction'.

3. A sequence element stops with time out because of - may be - busy data connection or something else. It is supposed to be restarted after having corrected the error. If the corresponding process is continued, the behavior depends on the critical section timeout handling strategy selected. If the 'Release lock ...' option is selected, the sequence element will directly run into an OOS condition because the expected sequence number has been incremented.

- OOS handling set to "Abort": sequence element fails with OOS condition. Process stops with error.
- OOS handling set to "Process as next": sequence element is processed. Expected sequence number is not changed.

If the 'Keep lock ...' option is selected, the sequence element can be reprocessed without problems, since the expected sequence number has not been changed.

4. If an error occurs inside the sequence flow, i.e. a file is not found, the process is stopped after error, the sequence element is set to 'Suspended' state, and the expected sequence number of the sequence is unchanged. The error can be fixed - here the missing file can be provided - and the process can be continued successfully.

5. If the sequence flow runs into a critical section timeout, the process is stopped with error, the appropriate sequence element is set to 'TimedOut' state. If the 'Release lock ...' strategy has been selected, the expected sequence number is incremented by 1. If the error is fixed, the process can only run successfully if OOS handling is set to 'Process as next'. Otherwise an OOS condition occurs and the process stops with error again.

1.24.4.3 Terminology and Definitions

Key word	Abbr.	Description
In-sequence processing		Kind of event sequencing. Processing incoming files/ messages either in receptio order or in ascending order of assigned sequence numbers.
Sequence		Process sequence. Reliable gap-free communication sequence for FIFO processing based on sequence numbers in monotonically ascending order relative to a sequence name.

Key word	Abbr.	Description
Element		Element of a process sequence. Identified by sequence name and sequence number.
Sequence name		Name of a process sequence as specified in the process/entity .
Golden transaction		The very first sequence number starting a new sequence.
Sequence number	SeqNo CurSeqNo	Generally positive Integer number. Allocation of sequence numbers is based on counters.
Sequence counter		Integer counter used for sequence number allocation. The names of sequence counters of FT adapters contain UUIDS of relevant master data records, e.g. partners or connections.
Expected sequence number	ExpSeqNo	A sequence element with this number has not yet been executed. The next process whose sequence number matches this one is allowed to enter the critical section.
Gap		Gap in the process sequence. The sequence number of the current process is greater than the expected sequence number of the associated sequence: `CurSeqNo > ExpSeqNo`.
Out of sequence	OOS	The out-of-sequence condition occurs if the sequence number of the current process is less than the expected sequence number of the associated sequence: `CurSeqNo < ExpSeqNo`.
Sequence number extraction		Extraction of the sequence number allocated to a process from the received message or file.
Critical section	CS	The sequence scope i.e. the section of the process that can be entered only once and that has to be executed indivisibly. Access is controlled by sequence name and sequence number. A process is only allowed to enter if its sequence name match the sequence name and its allocated sequence number fulfil expected sequence number condition of the critical section which is configured in the sequence profiles.. A sufficient but not necessary criterion for process parts to be inside a critical section is the need to access a shared resource.

Key word	Abbr.	Description
Mutual exclusion		Also called mutex or lock. At each point of time, at most one process may execute the critical section.
Monitor		Synchronization construct combining mutual exclusion with the ability to wait for a certain condition to become true.
EnterCriticalSection		Entry point of the critical section, trying section that each process must execute. Guarantees that only one process at a time is inside (critical section is locked). Controls each process that wants to enter. Each process has to wait until it acquires access.
LeaveCriticalSection		Exit point of the critical section. Executing it means to release the critical section, the expected sequence number is increased by 1 and the critical section is free again. The next process matching the sequence is allowed to enter.
Sequence profile		Process sequence configuration. Part of BIS GUI on configuration tab. Contains critical section timeout handling, sequence gap handling, out-of-sequence handling and instance-down handling as well as reorganization configuration.
Sequence flow		Special kind of sub-entity that is called by a sequence activity and is located one level below the calling entity (in the entity explorer).

1.24.5 Sequence Monitoring

Sequence monitoring GUI located on the Monitoring tab > Sequence processing > Sequence state. Table view showing all sequences:

Column name	Value	Description
Sequence name	*<SeqName>*[1]	The name of the sequence.
Sequence number	*<SeqNum>*[2]	The current sequence number. Its meaning depends on the *Sequenc e state*.
Sequence state		During sequence processing, several states are possible.
	Expecting	The sequence is currently not active. The *Sequence number* is the next expected sequence number.

Column name	Value	Description
	Running	There is currently a sequence element running inside the critical section.
	Timed out	Critical section timeout occurred. Timeout handling has been set to 'Keep lock ...'. Critical section is locked by the current sequence element. User interaction is necessary to release it.
Logical system	*<LS>*[3]	The logical system.
Active	True/False	Activation flag.
Last change	*<Timestamp>*[4]	Timestamp of last data change for this sequence.

1) Sequence name.
2) Sequence number.
3) Logical system.
4) Formatted timestamp.

For each sequence, it is possible to

- Display the sequence details (refer to sequence details table below).
- Skip the current sequence element (green ball icon in button bar):
  Skip the current sequence element and wait for the next sequence number.
- Skip all missing elements (yellow ball icon in button bar):
  Skip all missing elements and set the expected sequence number to the sequence number of the next available element.
- Abort the sequence (not available for Expecting states):
  Aborting the selected sequence element in running or pending state. The new state is 'Failed'.
- Pause sequence processing on all engines (red ball icon in button bar).
- Activate the selected sequence (available for inactive sequences).
- Deactivate the selected sequence (available for active sequences):
  All incoming sequence elements wait with pending state.

The sequence details is a table view that shows the history of the sequence. Visible are all sequence elements that are not reorganized.

Column name	Value	Description
Sequence number	*<SeqNum>*[1]	The current sequence number.
Sequence name	*<SeqName>*[2]	The name of the sequence.
Sequence state		The state of the sequence element.
	Finished	Final state. The sequence element has been processed successfully.
	Failed	Final state. Processing this sequence element has failed.
	Timed out	Final state. Processing this sequence element took too long, timeout occurred.

Column name	Value	Description
	Suspended	Non-final state. The sequence element has been suspended.
	Pending	Non-final state. The sequence element is waiting for its chance to enter the critical section.
	Running	Non-final state. The sequence element is being processed.
Sequence profiles	*<Ref>*[3]	The name of the sequence profile.
Last change	*<Timestamp>*[4]	Timestamp of last data change for this sequence element.
Timeout for leaving the critical section		The critical section timeout as determined by the sequence profile.
Process name		The name of the process.
Process ID		The process ID.
Instance		The BIS instance the sequence is assigned to.
Location path		The location path.
Logical system	*<LS>*[5]	The logical system.

1) Sequence number.
2) sequence name.
3) Reference to master data sequence profile.
4) Formatted timestamp.
5) Logical system.

For each sequence element, it is possible to

- Show the Process Inspector for the process details (green bullet icon in button bar).
- Abort the selected sequence element if it is not in final state, e.g. suspended or pending (blue square icon in button bar).

1.24.6 Examples and How-Tos

This sections contains some examples of use for several activities or parts of them.

1.24.6.1 How To: Bean Shell Scripts

1.24.6.1.1 Creating a BSH Script

1.24.6.2 Implementation

Each bean shell or BSH script must implement the `IScriptExecutor` interface :

```
package com.seeburger.scriptexecutor.base;

import com.seeburger.message.IAttachment;
import com.seeburger.script.exc.ScriptException;

import java.io.OutputStream;
import java.util.List;
import java.util.Map;

/**
 * IScriptExecutor defines the interface for calling scripts.
 * New version compared to {@link IExecutor}.
 */
public interface IScriptExecutor
{
    /**
     * @param clientId
     * @param version
     * @param scriptFile
     * @param attachments
     * @param inputParameters
     * @param outputAttachments
     * @param outputParameters
     * @param logStream
     * @return comment.
     * @throws ScriptException
     */
    String execute(String clientId, int version, String scriptFile, List attachments,
         Map inputParameters, List outputAttachments, Map outputParameters,
         OutputStream logStream) throws ScriptException;

    /**
     * Tries to kill the beanshell script and it's subprocess
     */
    void cancel();
}
```

This interface is always imported by default and must not be imported explicitly.

Example BSH script showing most important features:

- get input parameters
- set output parameters
- read input attachments
- write output attachments
- set debug levels
- write logfile

```
/*
 * ExampleScript01.java
 *
 * created at Aug 11, 2016
 *
 * Copyright (c) 2016 SEEBURGER AG, Germany. All Rights Reserved.
 */
package version1;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintStream;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.seeburger.message.IAttachment;
import com.seeburger.message.builder.AttachmentBuilder;
import com.seeburger.script.exc.ScriptException;

public class ExampleScript01
{
    /** hold the script's log, can be returned to the calling activity */
    static StringBuffer log = new StringBuffer(100);

    /** the debug level:
     *  generally, DEBUGLEVEL = 0 means no logging at all;
     *  DEBUGLEVEL = 1 means default logging;
     *  DEBUGLEVEL = 2 means more verbose logging */
    static int DEBUGLEVEL = 2;


    /**
     * @param clientId   as defined in process.
     * @param version   as defined in process.
     * @param scriptFile  as defined in process.
     * @param attachments  List<IAttachment> holding incoming attachments.
     * @param inputParameters  as defined in process.
     * @param outputAttachments List<IAttachment> holding created attachments.
     * @param outputParameters  for further access by process.
     * @param logStream  write logging messages to this stream.
     * @return comment
     * @throws ScriptException
     */
    String execute(String clientId,
                   int version,
         String scriptFile,
         List attachments,
         Map inputParameters,
         List outputAttachments,
         Map outputParameters,
         OutputStream logStream)
        throws ScriptException
    {
        trace("Starting ExampleScript01.bsh\n");

     // get input parameters
        String debugLevel = (String)inputParameters.get("DEBUG_LEVEL");
        DEBUGLEVEL = Integer.parseInt(debugLevel);
        trace("DEBUG_LEVEL is: " + debugLevel + "\n");


     // read input attachment
     StringBuffer inputContent = new StringBuffer();
        InputStream inputAttachment;
        try
        {
            IAttachment attachment = (IAttachment)attachments.get(0);
            inputAttachment = attachment.getContentAsStream();
            BufferedReader readerSrc = new BufferedReader(new
 InputStreamReader(inputAttachment));
     String line = null;
     while ((line = readerSrc.readLine()) != null)
     {
         inputContent = inputContent.append(line);
     }
        }
        catch (Exception e)
        {
            // always return the log after an error occurred
            PrintStream ps = new PrintStream(logStream);
            ps.println(log.toString());
            ps.println(e.getMessage());
            ps.close();

            // return error status
            outputParameters.put("STATUS", "ERROR");
            return "ERROR";
        }


     // set output parameters
     if (null == outputParameters)
```

```
     {
         outputParameters = new HashMap();
     }
     outputParameters.put("OUTPARAM1", "OutputParamTest");

     // create output attachment
        try
        {
            // create the attachment
         String outputText = "This is the content of the input attachment:\n" +
 inputContent.toString();
         AttachmentBuilder attachmentBuilder = AttachmentBuilder.getInstance();
         IAttachment attachment = attachmentBuilder.createFromBytes(outputText.getBytes(),
 "UTF-8");
            // add attachment to output list
         outputAttachments.add(attachment);
        }
        catch (Exception e)
        {
            // always return the log after an error occurred
            PrintStream ps = new PrintStream(logStream);
            ps.println(log.toString());
            ps.println(e.getMessage());
            ps.close();

            // return error status
            outputParameters.put("STATUS", "ERROR");
            return "ERROR";
        }

     // return the log if debug level is high enough
        if (DEBUGLEVEL >= 1)
        {
            PrintStream ps = new PrintStream(logStream);
            ps.println(log.toString());
            ps.close();
        }

        return "Script executed.";
    }


    /**
     *  Trace function. Writes to the script's log. Writes to console if DEBUGLEVEL is
 verbose.
     */
    private void trace(String s)
    {
        log.append(s);
        if (DEBUGLEVEL == 2)
        {
            System.out.print(s);
        }
    }

}
```

Script deployment and installation: refer to .

###

1.24.6.3 How To: Using scForEach

This section describes how to use the *scForEach* (page 10) activity.

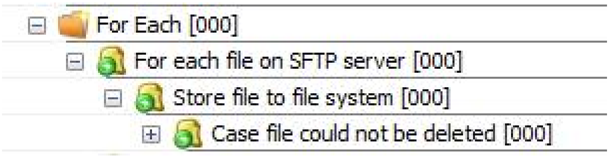
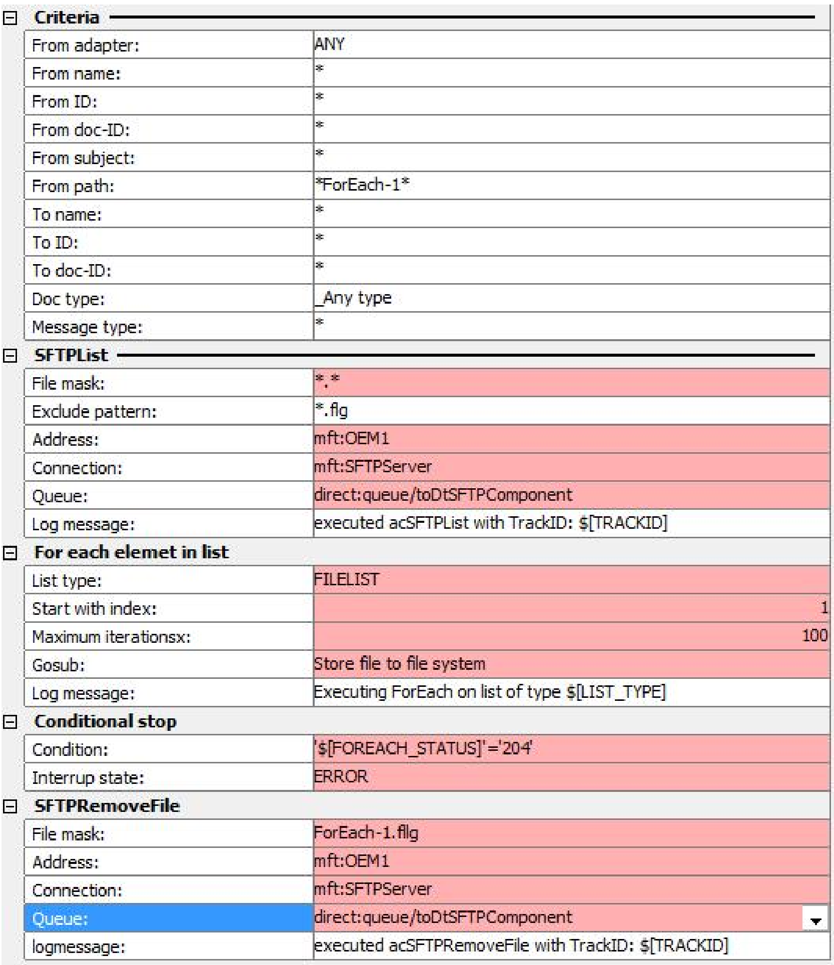
1.24.6.4 Scenario

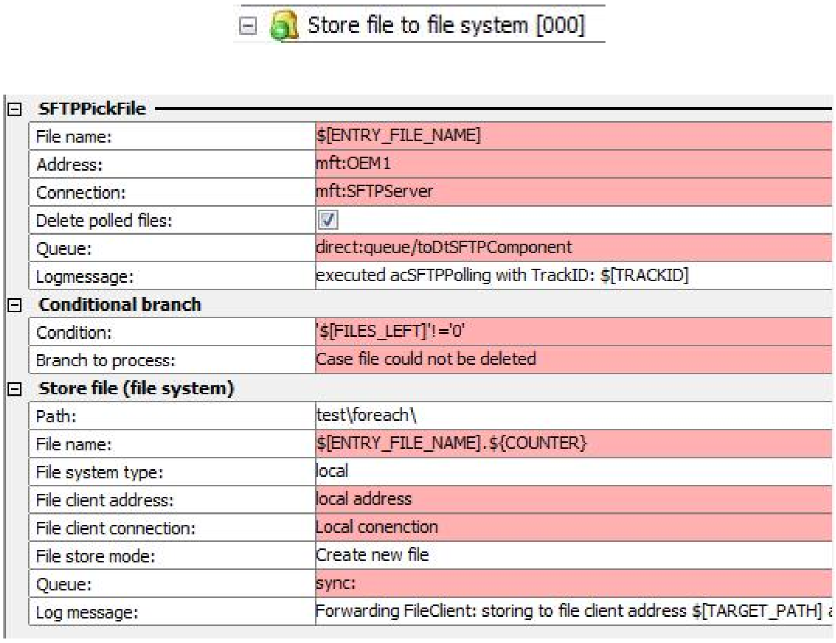
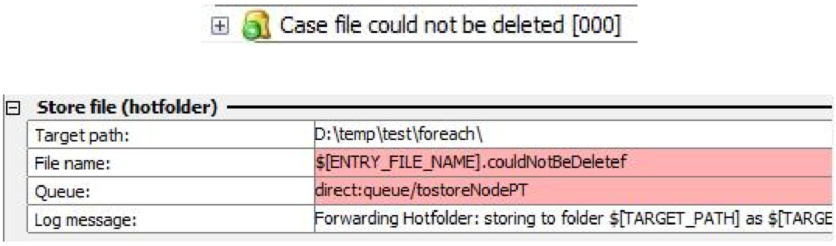
Files shall be fetched from a SFTP server and stored in a local directory with their original filenames.

Additional requirements:

a. An existing flag file shall start the operation.
b. The flag file shall be deleted after all files are successfully stored.
c. Files on SFTP server shall be deleted immediately after getting them in BIS.
d. In case of an error the flag file on SFTP server may not be deleted.
e. In case a file could not be deleted on SFTP server it may not be stored.

1.24.6.5 Implementation

Instruction	Illustration
1. The scenario above can be implemented by using a "ForEach"-Loop. Basic layout of TPM looks like this:	 ⊟ For Each [000] ⊟ For each file on SFTP server [000] ⊟ Store file to file system [000] ⊞ Case file could not be deleted [000]
2. The trigger is configured as scheduled SFTP polling to the flag file with “no delete” option. Configure SFTP connection and SFTP address in BIS master data and a scheduler which executes the polling. Assuming the flag file name is ForEach-1.flg. 3. In this example, a process started with a filename that matches *ForEach-1* starts a process with the configuration as described beside: The first activity in the process is a list to the SFTP folder, excluding the flag file pattern *.flg."SFTPList" returns a file list of found files on SFTP server. This list is processed in the "ForEach" activity, which is the next. "ForEach" executes the configured "GoSub" process for each element in this list. It is possible to set a start index and a maximum number of loops (in this example start index is 1 maximum 100 loops). Maximum is an upper limit. In this example, the sub process “Store to	 ⊟ For each file on SFTP server [000]  ⊟ **Criteria** From adapter: ANY From name: * From ID: * From doc-ID: * From subject: * From path: *ForEach-1* To name: * To ID: * To doc-ID: * Doc type: _Any type Message type: * ⊟ **SFTPList** File mask: *.* Exclude pattern: *.flg Address: mft:OEM1 Connection: mft:SFTPServer Queue: direct:queue/toDtSFTPComponent Log message: executed acSFTPList with TrackID: $[TRACKID] ⊟ **For each elemet in list** List type: FILELIST Start with index: 1 Maximum iterationsx: 100 Gosub: Store file to file system Log message: Executing ForEach on list of type $[LIST_TYPE] ⊟ **Conditional stop** Condition: '$[FOREACH_STATUS]'='204' Interrup state: ERROR ⊟ **SFTPRemoveFile** File mask: ForEach-1.flg Address: mft:OEM1 Connection: mft:SFTPServer Queue: direct:queue/toDtSFTPComponent logmessage: executed acSFTPRemoveFile with TrackID: $[TRACKID]

Instruction	Illustration
file system” is executed for each element in the list. After ForEach is finished, it is checked if an error happened during execution. The process stops and process status is set to "ERROR". If no error occurred, the flag file is removed via "SFTPRemoveFile".	
4. The first sub process gets the filename of the current entry in the list in property "ENTRY_FILE_NAME". Activity "SFTPPickFile" fetches this file. "SFTPPickFile" does not fail in case the file could not be deleted, but it reports this fact in the property "FILES_LEFT". "FILES_LEFT" must be 0 in case of a successful operation. The "ConditionalBranch" activity checks if "FILES_LEFT" is not 0 and branches to a further sub process in this case. Otherwise it stores the file.The sub process ends and "ForEach" calls it again with the next filename from the list.	 ⊟ Store file to file system [000] ⊟ **SFTPPickFile** File name: $[ENTRY_FILE_NAME] Address: mft:OEM1 Connection: mft:SFTPServer Delete polled files: ☑ Queue: direct:queue/toDtSFTPComponent Logmessage: executed acSFTPPolling with TrackID: $[TRACKID] ⊟ **Conditional branch** Condition: '$[FILES_LEFT]'!='0' Branch to process: Case file could not be deleted ⊟ **Store file (file system)** Path: test\foreach\ File name: $[ENTRY_FILE_NAME].${COUNTER} File system type: local File client address: local address File client connection: Local conenction File store mode: Create new file Queue: sync: Log message: Forwarding FileClient: storing to file client address $[TARGET_PATH] a
5. In case the file could not be deleted, the second sub process creates a file with the name of the file which could not be deleted and adds a ".couldNotBeDeleted" extension. Sending an email to an operator would be a typical alternative. After this activity, this sub process ends and "ForEach" continues with the next element in the list. The rest of the sub process “Store file to file system” is skipped.	 ⊞ Case file could not be deleted [000] ⊟ **Store file (hotfolder)** Target path: D:\temp\test\foreach\ File name: $[ENTRY_FILE_NAME].couldNotBeDeletef Queue: direct:queue/tostoreNodePT Log message: Forwarding Hotfolder: storing to folder $[TARGET_PATH] as $[TARGE

Instruction	Illustration
After the end of this sub process, the next activity is "ForEach".	

1.24.6.6 Final

It must be guaranteed that no second process is started by a polling while the process still runs.

Suggestion for an improvement:
The process is started with the flag file a payload. Let the first operation store the file with the name of the flag file. Delete this file after each is done, but before you stop the process with ERROR.

In case a second, intermediate polling tries to store the flag file it will fail. This solves the race condition.

1.24.6.7 How To: Splitting a Flat File

This section describes an example of using *acSplitFile* (page 127).

1.24.6.8 Scenario

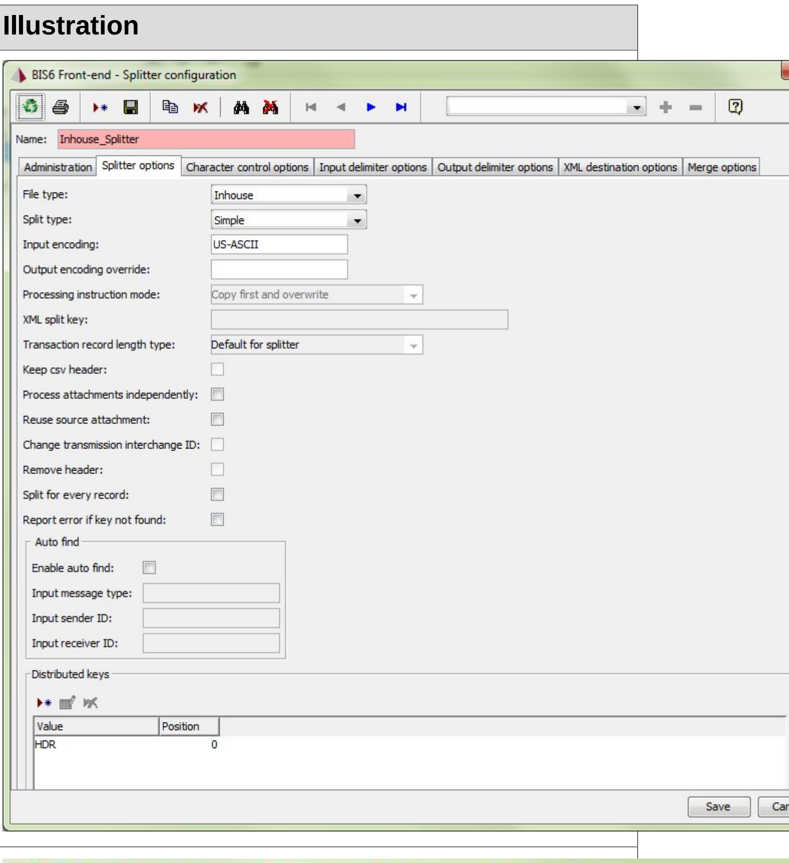
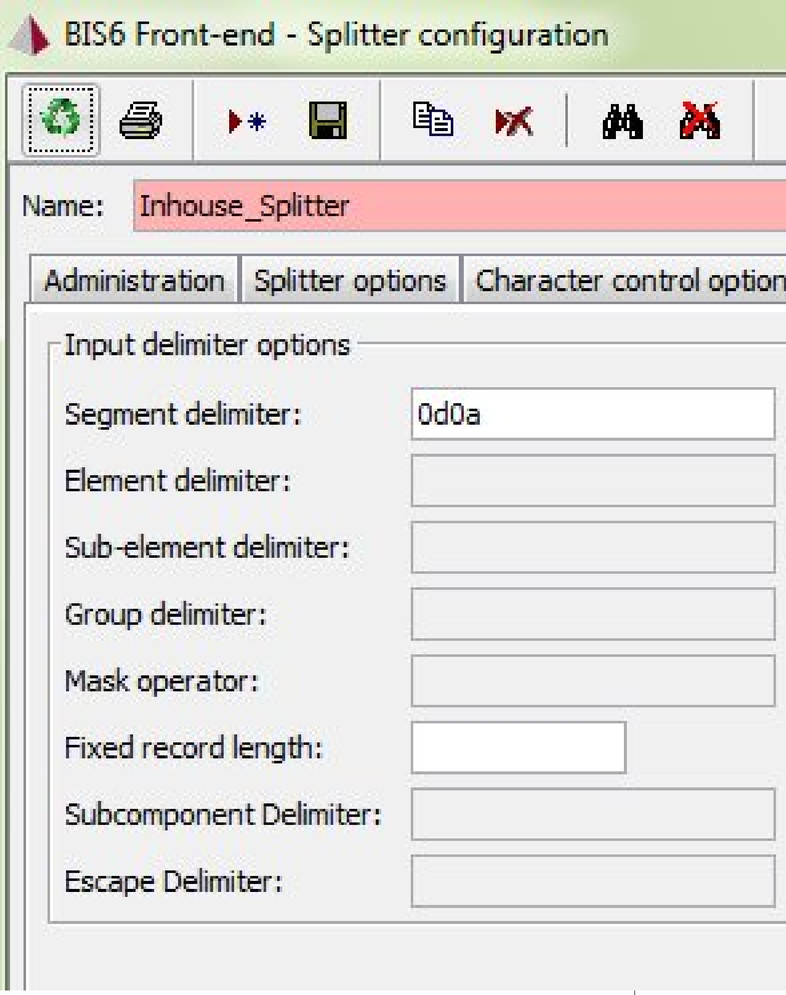
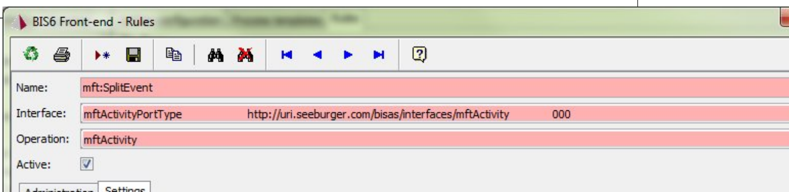
Example configuration for splitting a flat file.

File to be split:

```
HDROCED10722404
0160010722404,00000007279233,03122013,000000000185629,CONSOLIDATED ACCOUNT
….
0160010722404,00000007314675,03122013,000000000002000,CONSOLIDATED ACCOUNT
TRLOCED10722404
HDROCED10722405
0160010722404,00000007279233,03122013,000000000185629,CONSOLIDATED ACCOUNT
….
0160010722404,00000007314675,03122013,000000000002000,CONSOLIDATED ACCOUNT
TRLOCED10722405
```

1.24.6.9 Implementation

TODO

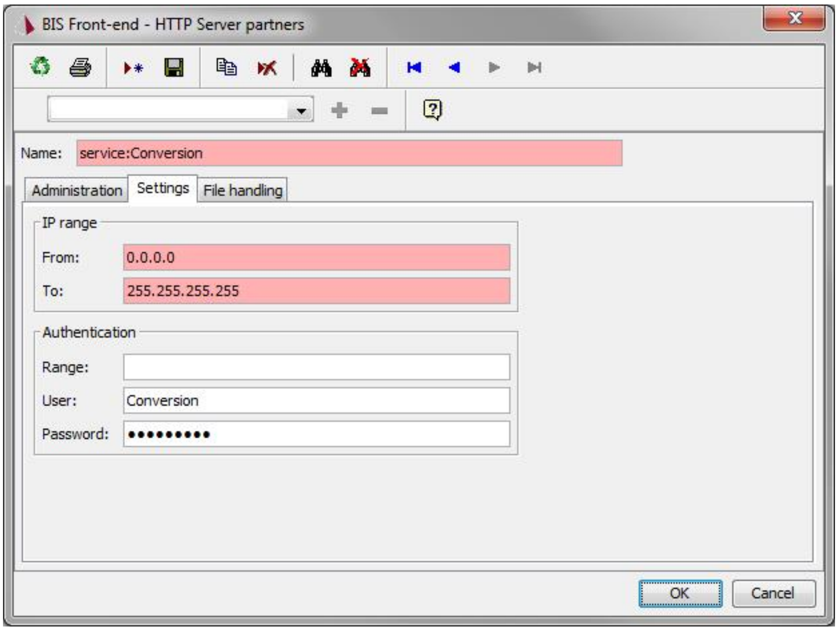
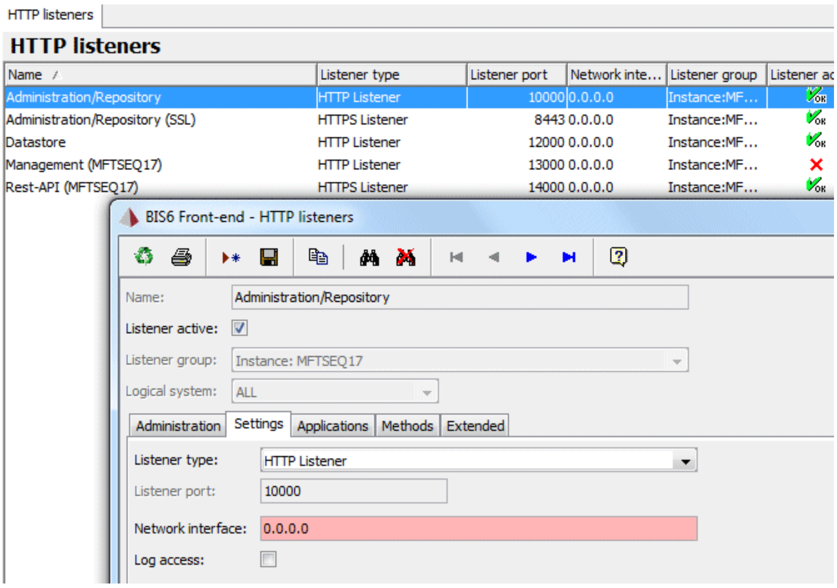
Instruction	Illustration
1. Create a new splitter configuration: - Enter the "Name". - Set "File type" to “Inhouse” - Create the distributed key “HDR” at position 0 – this is the beginning of a message in the source file .	 BIS6 Front-end - Splitter configuration Name: Inhouse_Splitter Administration | Splitter options | Character control options | Input delimiter options | Output delimiter options | XML destination options | Merge options File type: Inhouse Split type: Simple Input encoding: US-ASCII Output encoding override: Processing instruction mode: Copy first and overwrite XML split key: Transaction record length type: Default for splitter Keep csv header: □ Process attachments independently: □ Reuse source attachment: □ Change transmission interchange ID: □ Remove header: □ Split for every record: □ Report error if key not found: □ Auto find Enable auto find: □ Input message type: Input sender ID: Input receiver ID: Distributed keys Value: HDR, Position: 0 Save
2. Switch to the tab “Input delimiter options”: - Set the field “Segment delimiter” to “0D0A” (representing CRLF used as end of line).	 BIS6 Front-end - Splitter configuration Name: Inhouse_Splitter Administration | Splitter options | Character control option Input delimiter options Segment delimiter: 0d0a Element delimiter: Sub-element delimiter: Group delimiter: Mask operator: Fixed record length: Subcomponent Delimiter: Escape Delimiter:
3. Create a new rule: - Enter significant "Name". - Set the "Interface" to "mftActivityPortType" and "http://uri.seeburger.com/bisas/interfaces/mftActivity".	 BIS6 Front-end - Rules Name: mft:SplitEvent Interface: mftActivityPortType http://uri.seeburger.com/bisas/interfaces/mftActivity 000 Operation: mftActivity Active: ☑ Administration | Settings

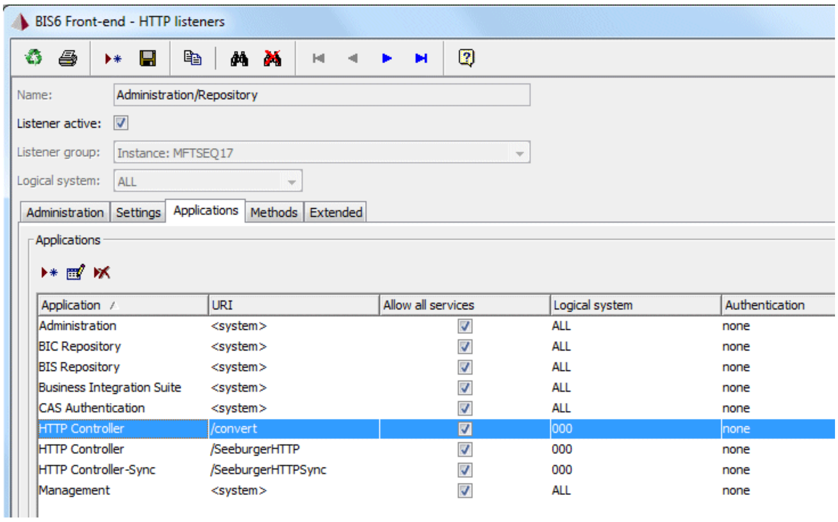
final

1.24.6.10 How To: HTTP Sync Reply

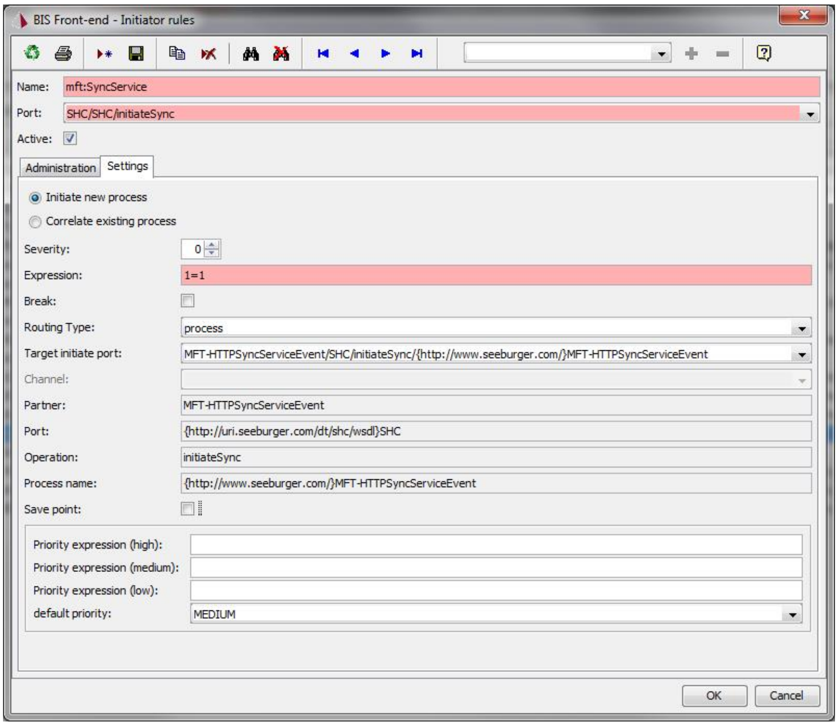
This section describes an example of using *acHTTPSyncReply* (page 238).

1.24.6.11 Sample Configuration

<table>
<tr><th>Instruction</th><th>Illustration</th></tr>
<tr><td>1. In "Configuration → Adapters → HTTP server → Partners" create a HTTP server user.</td><td>
BIS Front-end - HTTP Server partners
Name: service:Conversion
Administration | Settings | File handling
IP range
From: 0.0.0.0
To: 255.255.255.255
Authentication
Range:
User: Conversion
Password: •••••••••
OK Cancel</td></tr>
<tr><td>2. Go to "Configuration → System settings → HTTP services → HTTP listeners".</td><td>
HTTP listeners
<b>HTTP listeners</b>
<table>
<tr><th>Name</th><th>Listener type</th><th>Listener port</th><th>Network inte…</th><th>Listener group</th><th>Listener ac</th></tr>
<tr><td>Administration/Repository</td><td>HTTP Listener</td><td>10000</td><td>0.0.0.0</td><td>Instance:MF…</td><td>✓</td></tr>
<tr><td>Administration/Repository (SSL)</td><td>HTTPS Listener</td><td>8443</td><td>0.0.0.0</td><td>Instance:MF…</td><td>✓</td></tr>
<tr><td>Datastore</td><td>HTTP Listener</td><td>12000</td><td>0.0.0.0</td><td>Instance:MF…</td><td>✓</td></tr>
<tr><td>Management (MFTSEQ17)</td><td>HTTP Listener</td><td>13000</td><td>0.0.0.0</td><td>Instance:MF…</td><td>✗</td></tr>
<tr><td>Rest-API (MFTSEQ17)</td><td>HTTPS Listener</td><td>14000</td><td>0.0.0.0</td><td>Instance:MF…</td><td>✓</td></tr>
</table>
BIS6 Front-end - HTTP listeners
Name: Administration/Repository
Listener active: ☑
Listener group: Instance: MFTSEQ17
Logical system: ALL
Administration | Settings | Applications | Methods | Extended
Listener type: HTTP Listener
Listener port: 10000
Network interface: 0.0.0.0
Log access: □</td></tr>
</table>

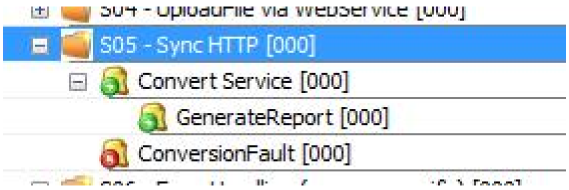
<table>
<tr><th>Instruction</th><th>Illustration</th></tr>
<tr><td>3. Go to the Applications tab. Create a new application with a path – should be the service name – here “/ convert”.</td><td>
BIS6 Front-end - HTTP listeners
Name: Administration/Repository
Listener active: ☑
Listener group: Instance: MFTSEQ17
Logical system: ALL
Administration | Settings | Applications | Methods | Extended
Applications
<table>
<tr><th>Application</th><th>URI</th><th>Allow all services</th><th>Logical system</th><th>Authentication</th></tr>
<tr><td>Administration</td><td><system></td><td>☑</td><td>ALL</td><td>none</td></tr>
<tr><td>BIC Repository</td><td><system></td><td>☑</td><td>ALL</td><td>none</td></tr>
<tr><td>BIS Repository</td><td><system></td><td>☑</td><td>ALL</td><td>none</td></tr>
<tr><td>Business Integration Suite</td><td><system></td><td>☑</td><td>ALL</td><td>none</td></tr>
<tr><td>CAS Authentication</td><td><system></td><td>☑</td><td>ALL</td><td>none</td></tr>
<tr><td>HTTP Controller</td><td>/convert</td><td>☑</td><td>000</td><td>none</td></tr>
<tr><td>HTTP Controller</td><td>/SeeburgerHTTP</td><td>☑</td><td>000</td><td>none</td></tr>
<tr><td>HTTP Controller-Sync</td><td>/SeeburgerHTTPSync</td><td>☑</td><td>000</td><td>none</td></tr>
<tr><td>Management</td><td><system></td><td>☑</td><td>ALL</td><td>none</td></tr>
</table></td></tr>
<tr><td>4. Create a new rule configuration that catches any http request sent to the service. May look like this in case a specific URL was created to identify these requests.

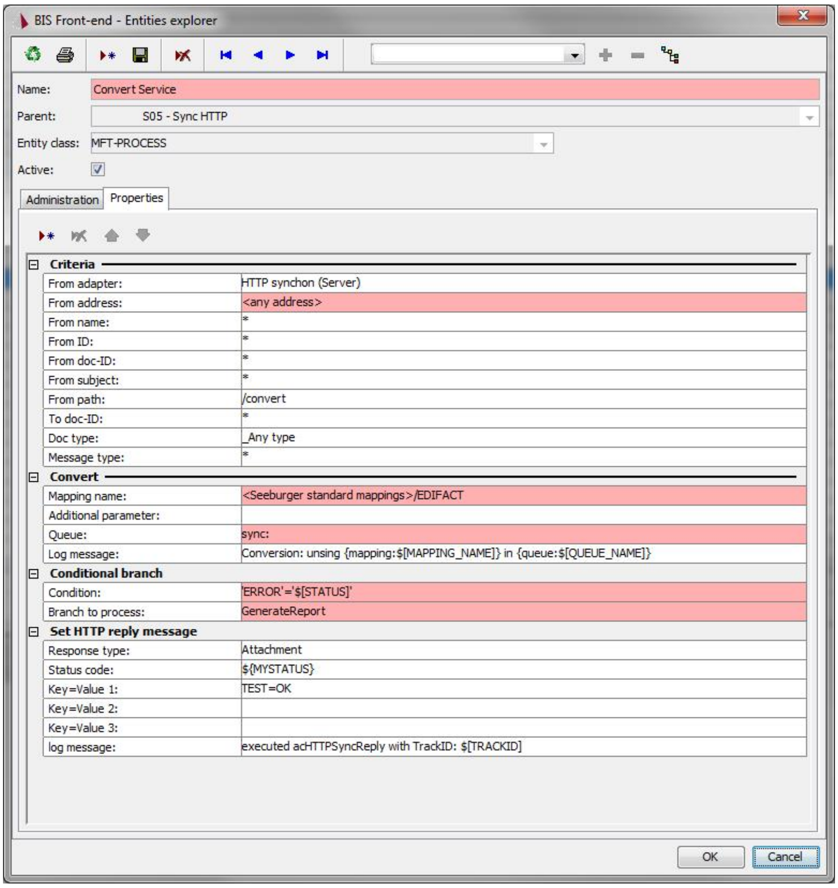
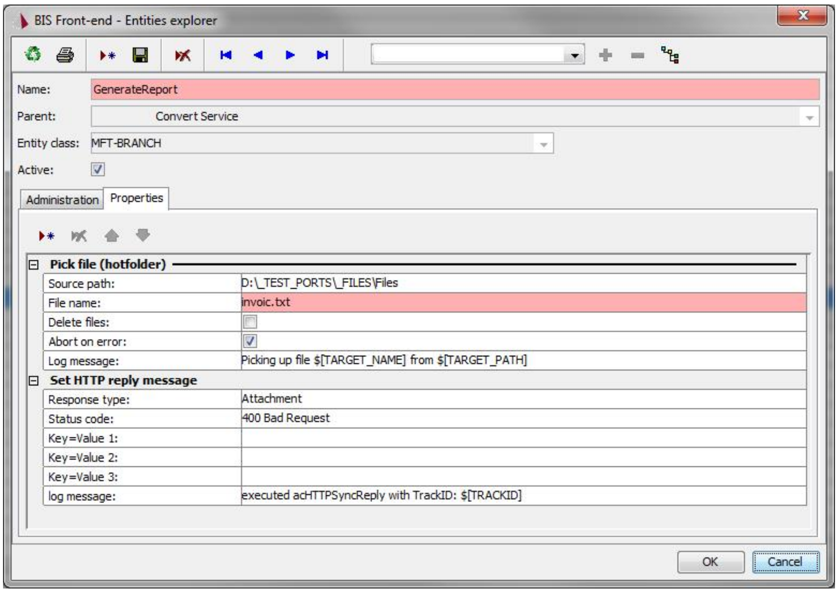
The expression can be set to 1=1 if all synchronous requests will be handled by MFT-Solution. Otherwise set //requestURI='/convert'.

Disable “Save point” because the save point is useless in case of synchronous requests and consumes system resources.</td><td>
BIS Front-end - Initiator rules
Name: mft:SyncService
Port: SHC/SHC/initiateSync
Active: ☑
Administration | Settings
● Initiate new process
○ Correlate existing process
Severity: 0
Expression: 1=1
Break: □
Routing Type: process
Target initiate port: MFT-HTTPSyncServiceEvent/SHC/initiateSync/{http://www.seeburger.com/}MFT-HTTPSyncServiceEvent
Channel:
Partner: MFT-HTTPSyncServiceEvent
Port: {http://uri.seeburger.com/dt/shc/wsdl}SHC
Operation: initiateSync
Process name: {http://www.seeburger.com/}MFT-HTTPSyncServiceEvent
Save point: □
Priority expression (high):
Priority expression (medium):
Priority expression (low):
default priority: MEDIUM
</td></tr>
</table>

Setting up the TPM

In this example, a specific use case is shown: “Providing a conversion service which replies a http fault in case of conversion fault."

<table>
<tr><th>Instruction</th><th>Illustration</th></tr>
<tr><td>1. Overall TPM layout</td><td>
S04 - UploadFile via WebService [000]
S05 - Sync HTTP [000]
Convert Service [000]
GenerateReport [000]
ConversionFault [000]</td></tr>
</table>

Instruction	Illustration
2. Entity Convert Service. Set the Queue to “sync:” to get fastest response.	
3. Entity "GenerateReport".	

BIS Front-end - Entities explorer

Name:	Convert Service
Parent:	S05 - Sync HTTP
Entity class:	MFT-PROCESS
Active:	☑

Administration | Properties

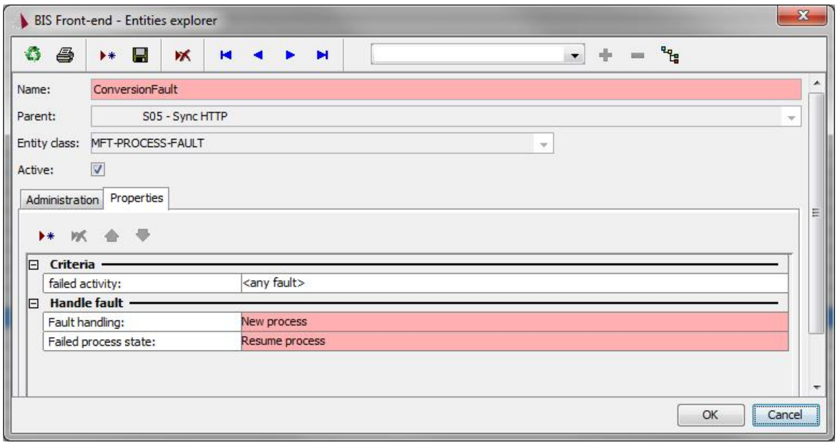
Criteria	
From adapter:	HTTP synchon (Server)
From address:	<any address>
From name:	*
From ID:	*
From doc-ID:	*
From subject:	*
From path:	/convert
To doc-ID:	*
Doc type:	_Any type
Message type:	*
Convert	
Mapping name:	<Seeburger standard mappings>/EDIFACT
Additional parameter:	
Queue:	sync:
Log message:	Conversion: unsing {mapping:$[MAPPING_NAME]} in {queue:$[QUEUE_NAME]}
Conditional branch	
Condition:	'ERROR'='$[STATUS]'
Branch to process:	GenerateReport
Set HTTP reply message	
Response type:	Attachment
Status code:	${MYSTATUS}
Key=Value 1:	TEST=OK
Key=Value 2:	
Key=Value 3:	
log message:	executed acHTTPSyncReply with TrackID: $[TRACKID]

BIS Front-end - Entities explorer

Name:	GenerateReport
Parent:	Convert Service
Entity class:	MFT-BRANCH
Active:	☑

Administration | Properties

Pick file (hotfolder)	
Source path:	D:_TEST_PORTS_FILES\Files
File name:	invoic.txt
Delete files:	□
Abort on error:	☑
Log message:	Picking up file $[TARGET_NAME] from $[TARGET_PATH]
Set HTTP reply message	
Response type:	Attachment
Status code:	400 Bad Request
Key=Value 1:	
Key=Value 2:	
Key=Value 3:	
log message:	executed acHTTPSyncReply with TrackID: $[TRACKID]

Instruction	Illustration
4. Entity "ConversionFault".	 BIS Front-end - Entities explorer Name: ConversionFault Parent: S05 - Sync HTTP Entity class: MFT-PROCESS-FAULT Active: ☑ Administration | Properties Criteria failed activity: <any fault> Handle fault Fault handling: New process Failed process state: Resume process OK | Cancel

1.24.6.12 Testing

This configuration can be tested, for example with SoapUI. Simple test witha web browser does not succeed, because the process requires an attachment.

SoapUI can be downloaded from the internet.

After started, any available WSDL file can be loaded. Open the request of any operation. SoapUI will show the specific request message – replace the content with any text – e.g the file you want to convert.

On top of the request windows, a url is shown – left mouse click – [edit current …] change the URL to http://localhost:10000/convert?ls=000&HTTPInterface=syncReply&MyProp=test

On the bottom of the same window is a small button “Aut” – click on it and insert user name/password as set in the HTTP server address field in BIS GUI.

Execute test by clicking the small green arrow on top of the request window.

1.24.6.13 How To: In-Sequence Processing

Requirements

- User rights for GUI: 'SHOW_SEQUENCE' and 'SEQUENCE_EDIT_ALLOWED'.

Allocation and Retrieval of sequence numbers

In-sequence processing as supported by adapters means to process the received messages in the order of their time of reception. This actually means FIFO processing. The sample configurations below cover the following adapters which allocate the sequence numbers in reception order of the messages:

- HTTP (HTTP partner address allows sequencing to be switched on and off)
- JMS (JMS connection allows sequencing to be switched on and off)

Each adapter creates a counter that matches the configured sequence counter pattern. Sequence number retrieval inside the process that handles the received message is provided by the following XPath expression: "$(see:getContextProperty('SEQUENCE_NUMBER'))". The name of the used sequence counter is also provided by the adapter: "$(see:getContextProperty('SEQUENCE_NAME))". These sequence counters are created by the adapters and can be displayed in the GUI - Configuration > System settings > Counters.

Another type of sequencing does not need supporting adapters. The sending component is responsible for allocating the sequence number to the file (attachment) or message to be sent. The receiving component is responsible for retrieving this sequence number from the attachment or message. In case of direct sequence number retrieval from an attachment, *acContentAnalyzer* (page 63) and *acConverter* (page 63) activities can be used.

1.24.6.13.1 Sample Configuration HTTP (FIFO)

Setting up HTTP

Includes configuration of the sequence number allocation using the HTTP adapter. When receiving a file via HTTP the HTTP controller allocates the sequence number to the incoming HTTP message using the sequence counter pattern specified in the HTTP partner address (Refer to Partner addresses in the HTTP controller manual (link) for more details). Steps 2-4 create a simple entity which allows testing by sending a file via HTTP.

1. HTTP Server Partner	• Go to Configuration ⇒ Adapters ⇒ HTTP Server ⇒ Partners. • Select the partner or create a new one. • Go to sequence processing tab.	• Tick "Prepare messages for in sequence processing" checkbox. • Adapt sequence counter pattern: Default counter pattern is 'SEQ_HTTP_CONTROLLER_{PARTNER_U Extendable by using the content type: 'SEQ_HTTP_CONTROLLER_{PARTNER_U Also extendable by enabling the HTTP header name: 'SEQ_HTTP_CONTROLLER_{PARTNER_U

Setting up the Sequence

Includes sequence profile configuration (refer to In-sequence processing (page 400) for more details about sequence profiles) and use of *scSequence* (page 36) activity.

1. Sequence profile	• Go to Configuration ⇒ Sequence processing ⇒ Sequence profiles	• Name := "RELEASE_GAPSKIPALL_OOSABORT"
	• Go to Settings tab • Configure critical section timeout handling	• Timeout for leaving critical section := 65 seconds • Timeout handling := "Release lock and abort process with error"
	• Configure sequence gap handling	• Max allowed gap := 5 • Timeout for resolving gap := 120 seconds • Timeout handling for gap := "Skip all missing elements"
	• Configure out-of-sequence handling	• Handling of oos elements := "Abort process with error"
	• Configure instance-down handling	• Handling if instance is down := "Wait until the instance is back"

	• Configure sequence reorganization	• Reorganize sequence := "Do not reorganize"
2. Main entity	• Go to Configuration ⇒ Entities explorer ⇒ All entities.	• Create a new entity that will receive the file sent by the entity above.
3. HTTP criteria	• Create the criteria	• From adapter := "HTTP asynchronous (Server)" • From address := <select HTTP server partner>
	• Save the entity	
4. Sequence flow entity	• Create a sequence flow sub-entity using entity class "MFT-SEQUENCE". • Add *acSetInfo* (page 45) activity.	• Info 14 := "$(see:getContextProperty('SEQUENCE_NU
	• Store sequence flow entity	
5. Main entity	• Reopen previous main entity	
6. Sequence activity	• Add the *scSequence* (page 36) activity	• Profile := <select sequence profile above> • Name of sequence := "HTTPServerEventAsync01" • Sequence number : = "$(see:getContextProperty('SEQUENCE_NU • Sequence flow := <select sequence flow entity above>

Setting up the Test

The test entity is just a simple entity with hotfolder criteria that sends an HTTP message.

1. HTTP send entity	• Go to Configuration ⇒Entities explorer ⇒ All entities. • Create a new entity that will send a file via HTTP.	• Name := "HTTP_Send_01" • Active = "TRUE"
2. Hotfolder criteria	• Configure criteria (here hotfolder file event)	• From adapter := "File event (hotfolder)" • From address := "mft:" • From path := "HTTP_Send_01"
3. HTTP send activity	• Add the *mdHTTP* (page 241) activity.	• Master data usage type := "Only master data" • Queue := "direct:" • Send method := "PUT" • Receiver := <select HTTP Client address> • Connection := <select HTTP Client connection> • Proxy mode := "DT_PROXY_MODE_NONE">

Create a file whose name matches the hotfolder criteria and put it into the specified hotfolder directory.

1.24.6.13.2 Sample Configuration JMS (FIFO)

Setting up JMS

Configuration of the sequence number allocation using the JMS adapter. When receiving a JMS message the JMS adapter allocates the sequence number to the incoming JMS message using the sequence counter pattern specified in the referenced JMS connection (refer to JMS connection settings in the JMS Adapter manual (link) for more details).

1. JMS connection	• Go to Configuration ⇒ Adapters ⇒ JMS Client ⇒ Connections. • Select the connection or create a new one. • Go to sequence processing tab.	• Tick "Prepare messages for in sequence processing" checkbox. • Sequence counter pattern cannot be adapted: 'SEQ_JMS_CLIENT_{CONNECTION-UUID}_{ADDRESS-UUID}'.

Setting up the Sequence

Includes sequence profile configuration (refer to Sequence processing (page 400) for more details about sequence profiles) and use of *scSequence* (page 36) activity. Refer to **Sample Configuration HTTP** above about the sequence profile sample configuration.

1. Main entity	• Go to Configuration ⇒ Entities explorer ⇒ All entities. • Create a new entity.	• Name := "JMSEvent01" • Active = "TRUE"
2. JMS criteria	• Create the criteria	• From adapter := "JMS event" • From address := <select JMS client address>
	• Save the entity	
3. Sequence flow entity	• Create a sequence flow sub-entity using entity class "MFT-SEQUENCE". • Add *acSetInfo* (page 45) activity.	• Info 14 := "$(see:getContextProperty('SEQUENCE_NU
	• Save the entity	
4. Main entity	• Reopen previous main entity	
5. Sequence activity	• Add the *scSequence* (page 36) activity	• Profile := <select sequence profile above> • Name of sequence := "JMSEvent01" • Sequence number : = "$[SeqNum]" • Sequence flow := <select sequence flow entity above>

Setting up the Test

The test entity is just a simple entity with hotfolder criteria that sends a JMS message to the queue and a second poll entity that picks a single JMS message from the queue.

1. JMS send entity	• Go to Configuration ⇒Entities explorer ⇒ All entities. • Create a new entity that will send a file via JMS.	• Name := "JMS_Send_Text01" • Active = "TRUE"
2. Hotfolder criteria	• Configure criteria (here hotfolder file event)	• From adapter := "File event (hotfolder)" • From address := "mft:" • From path := "JMS_Send_Text01"
3. JMS send activity	• Add the *mdJMS* (page 254) activity.	• Receiver := <select JMS Client Address> • Connection := <select JMS Client Connection> • Message type := "text" • Delivery mode := "<default>" • Abort on error := "TRUE" • Queue := "direct:queue/ toDtJMSComponent"
4. Poll entity	• Create new entity	• Name := "JMS_PickFile_NoID01" • Active = "TRUE"
5. Hotfolder criteria	• Configure criteria (here hotfolder file event)	• From adapter := "File event (hotfolder)" • From address := "mft:" • From path := "JMS_PickFile_NoID01"
6. Poll activity	• Add the *acJMSPickFile* (page 252) activity.	• Receiver := <select JMS Client Address from test entity> • Connection := <select JMS Client Connection from test entity> • Get by ID := "FALSE" (switch it off) • Abort on error := "TRUE" • Queue := "direct:queue/ toDtJMSComponent"

Create a file whose name matches the hotfolder criteria and put it into the specified hotfolder directory.

1.24.6.13.3 Sample Configuration with Sequence Number Extraction from Attachment

Setting up the Sequence

Includes sequence profile configuration (refer to Sequence processing (page 400) for more details about sequence profiles) and use of *scSequence* (page 36) activity. Refer to **Sample Configuration HTTP** above about the sequence profile sample configuration.

This sample configuration uses a content analyzer that extracts the sequence number from the file received by a hotfolder event (refer to BIC File Content Extractor manual (link) for details about how to configure a content analyzer). Another possibility is to use a mapping.

1. Content Analyzer	• Go to Configuration ⇒ ######### ⇒ ######### ############# • ###### ### ######### #############	• Name := "SeqNumExtraction" Output property: $[SeqNum] with extracted sequence number.
2. Main entity	• Go to Configuration ⇒ Entities explorer ⇒ All entities. • Create a new entity.	• Name := "Hotfolder_Receive01" • Active = "TRUE"
3. Hotfolder criteria	• Create the criteria	• From adapter := "Hotfolder event" • From address := "mft:"
4. ContentAnalyzer activity	• Add the *acContentAnalyzer* (page 63) activity	• Analyzer configuration := "SeqNumExtraction"
	• Save the entity	
5. Sequence flow entity	• Create a sequence flow sub-entity using entity class "MFT-SEQUENCE". • Add *acSetInfo* (page 45) activity.	• Info 14 := "$[SEQUENCE_NUMBER]".
	• Save the entity	
6. Main entity	• Reopen previous main entity	
7. Sequence activity	• Add the *scSequence* (page 36) activity	• Profile := <select sequence profile above> • Name of sequence := "Hotfolder_Receive_01" • Sequence number : = "$[SeqNum]" • Sequence flow := <select sequence flow entity above>

Setting up the Test

Create a file whose name matches the hotfolder criteria and put it into the specified hotfolder directory.

1.24.6.13.4 Resetting a Sequence

Sequence resetting might be necessary when there occurs a sequence number overflow when using sequence number extraction from an attachment (e.g. 4-digit format of sequence number in attachment reaches '9999', next sequence number will be '0' or '1'). If this kind of rotation is necessary or desired, the sequence name has to be provided in the attachment, too:

If the last sequence element of the "old" sequence is 'SeqRotation01 / 9999'. The next sequence element that arrives should be 'SeqRotation02 / 1'. This restarts the sequence. If reorganization is configured in the sequence profile the "old" sequence will be reorganized and thus deleted after some time. This example uses a counter for sequence resetting. It is also possible to achieve a daily rotation by using the current date as part of the sequence name (for instance by using XPath expression (page 392) '$(seeutil:getCurrentDate('yyyyMMdd'))').

Setting up the Sequence

Includes sequence profile configuration (refer to Sequence processing (page 400) for more details about sequence profiles) and use of *scSequence* (page 36) activity. Refer to **Sample Configuration HTTP** above about the sequence profile sample configuration.

This sample configuration uses a content analyzer that extracts the sequence number and sequence name from the file received by a hotfolder event (refer to BIC File Content Extractor manual (link) for details about how to configure a content analyzer). Another possibility is to use a mapping.

1. Content Analyzer	• Go to Configuration ⇒ ######### ⇒ ######### ############# • ###### ### ######### #############	• Name := "SeqNumAndNameExtraction" Output properties: • $[SeqNum] with extracted sequence number. • $[SeqName] with extracted sequence name.
2. Main entity	• Go to Configuration ⇒ Entities explorer ⇒ All entities. • Create a new entity.	• Name := "Hotfolder_Receive01" • Active = "TRUE"
3. Hotfolder criteria	• Create the criteria	• From adapter := "Hotfolder event" • From address := "mft:"
4. ContentAnalyzer activity	• Add the *acContentAnalyzer* (page 63) activity	• Analyzer configuration := "SeqNumAndNameExtraction"
	• Save the entity	
5. Sequence flow entity	• Create a sequence flow sub-entity using entity class "MFT-SEQUENCE". • Add *acSetInfo* (page 45) activity.	• Info 14 := "$[SEQUENCE_NUMBER]". • Info 15 := "$[SEQUENCE_NAME]".
	• Save the entity	
6. Main entity	• Reopen previous main entity	
7. Sequence activity	• Add the *scSequence* (page 36) activity	• Profile := <select sequence profile above> • Name of sequence := "$[SeqName]" • Sequence number : = "$[SeqNum]" • Sequence flow := <select sequence flow entity above>

Setting up the Test

Create a file whose name matches the hotfolder criteria and whose content matches the format required by the content analyzer. Put it into the specified hotfolder directory.

1.24.6.14 How To: CMA - Introduction

1.24.6.14.1 General

This how-to is based on the MFT solution and AS2 REST data.

CMA instance: creation, execution, monitoring of onboarding campaigns and update services, connected to a BIS instance
BIS instance: load REST data for CMA form initialization, validate EDIFACT messages, store entered/collected data

Onboarding campaign: based on a CMA survey, offers the following features

- Form initialization:
  Load REST data from a connected BIS instance in order to initialize the CMA form with default data.
- Compliance check:
  Send a compliance check request to a connected BIS instance for EDIFACT compliance checks.
- Data export:
  Stored the collected data to a connected BIS instance as new REST data.

Update service: based on a CMA self-service, offers the following features

- Form initialization:
  Load the REST data for the self-service user from a connected BIS instance in order to initialize the CMA form with the data stored during the last update session.
- Compliance check:
  Send a compliance check request to a connected BIS instance for EDIFACT compliance checks.
- Data export:
  Stored the collected data to a connected BIS instance, create/update REST data for the self-service user.

1.24.6.14.2 Requirements

CMA instance:

- CMA account with sufficient rights for creating campaign templates as well as creating, launching and monitoring campaigns.
- Connection to BIS instance for loading and storing REST data.
- Connection to BIS instance for EDIFACT compliance checks.

BIS instance:

- MFT solution installed including S-18000-CMA.
- AS2 REST GUI and service installed: BIS652 SP42 or higher.
- MTP tool installed.
- ScriptExecutor started (needed by S-18000-CMA activities).
- WebService Controller installed and started (if needed, in cases of form initialization and data export via SOAP requests or compliance checks).
- HTTP Client and Controller installed and started (if needed, in cases of form initialization and data export via HTTP requests).
- Edifact Compliance adapter installed and started (if needed, in cases of compliance check).

1.24.6.14.3 Main Steps

1. Get an AS2 CMA form or generate it from the AS2 REST form used in BIS.
2. Create a CMA template and import the form. Check and adapt the form.
3. Create a CMA survey campaign or self-service project from the template. Adapt the form according to this campaign or self-service.
4. Configure form initialization.
5. Configure EDIFACT compliance check (if needed).
6. Configure the data export that will store the gathered data on target BIS instance.

1.24.6.14.4 How To: CMA Form Generation

Task:
Convert the AS2 REST form (generic form) into a CMA form.

Requirements BIS installation:

- AS2 REST GUI and service installed: BIS653 SP42 or higher
- S-11001-LIB installed

Steps:

1. Determine form name
2. Create HTTP server partner
3. Create HTTP listener or use an existing one
4. Create matching rule
5. Create and send HTTP request

Determining the Name of the Form

SeeConfig file provides ionformation about all generic forms used in the BIS installation.

Location: <BIS>\conf\SeeConfig\{com.seeburger.conf.user=BISAS}\{com.seeburger.conf=system}

Open com.seeburger.menutree-config.xml:
Look for `<node name="trnEndpoints">` denoting the Endpoints menu entry in the frontend.

```
<node name="trnEndpoints">
    <property name="image">closedfolder.gif</property>
    <property name="permission">SHOW_ENDPOINTS</property>
    <property name="label">CONF.ENDPOINTS</property>
    <property name="priority" type="java.lang.Interger">25</property>
    <property name="translate" type="java.lang.Boolean">true</property>
    <property name="dirname">ENDPOINTs</property>
    <property name="trnMasterDataAS2REST">
        <property name="image">closedfolder.gif</property>
        <property name="permission">as2-rest:view-masterdata</property>
        <property name="label">AS2</property>
        <property name="priority" type="java.lang.Interger">1</property>
        <property name="translate" type="java.lang.Boolean">true</property>
        <property name="dirname">AS2</property>
        <property name="trnMasterDataAS2RESTPartners">
            <property name="image">InsTerminatedPanel16.gif</property>
            <property name="module">MasterDataAS2RESTPartnersModule</property>
            <property name="permission">as2-rest:view</property>
            <property name="image">Partners</property>
            <property name="priority" type="java.lang.Interger">1</property>
            <property name="translate" type="java.lang.Boolean">true</property>
        </property>
```

Get the name of the module: MasterDataAS2RESTPartnersModule.

Open com.seeburger.module-config.xml:

```
<node name="MasterDataAS2RESTPartnersModule">
   <property name="runnable" type="java.langBoolean">false</property>
   <property name="multipleEx" type="java.langBoolean">false</property>
   <property name="loadonstartup" type="java.langBoolean">false</property>
   <property name="multiple" type="java.langBoolean">false</property>
   <property name="exclusive" type="java.langBoolean">false</property>
   <property name="nlsresource">com.seeburger.as2.nls.AS2RestMasterData</property>
   <property name="permission"as2-rest:view</property>
   <property name="class"com.seeburger.as2.gui.modules.AS2PartnerRestModule</property>
    <node name="label">
       <property name="text">as2User.AS2Partners</property>
       <property name="translate" type="java.langBoolean">false</property>
   </node>
   <node name="gui">
       <property name="closeable" type="java.langBoolean">false</property>
       <property name="movable" type="java.langBoolean">false</property>
       <property name="focusable" type="java.langBoolean">false</property>
       <property name="intialPos">Center</property>
       <property name="selected" type="java.langBoolean">false</property>
   </node>
   <node name="kvpair">
       <property name="ListQueryParam">as2id=*</property>
       <property name="Perm-Create"></property>
       <property name="IDPath">service.@as2Id + '/' + @as2Id + '?show-secured=true'</
property>
       <property name="FormDef">frmAS2RestPartner.xml</property>
       <property name="Perm-Delete">as2-rest:delete</property>
       <property name="URLPath">/as2</property>
       <property name="Perm-Edit">as2-rest:modify</property>
       <property name="EnableMTP">fals</property>
   </node>
```

Look for the node whose name is mathcing the module name from above. Get the form name: frmAS2RestPartner.xml.

Form location:
<BIS>/data/webdav/repository/conf/forms.

Preparing the Conversion

1. Create an HTTP server partner:
   IP range - From := IP of BIS instance
   IP range - To := IP of BIS instance
2. Create an HTTP listener for a synchronous request or use an existing one:
   Listener type := HTTP Listener
   Listener port := 18080
   Application := HTTP Controller-Sync
   URI := /cmahttp/*
3. Create a rule:
   Name := APIFormGenerator
   Interface := SHC
   Operation := initiateSync
   Initiate new process := true
   Expression := //queryStringList[//key='mode']/value='CMAFormConverter'
   Routing Type := process

Process/partner := see:API_CMAFormGenerator/API_CMAFormGenerator/

Form Conversion

Enter the following URL in a browser:

http://<IP_BIS>[1]:<Port_HTTPListener>[2]/<URI_HTTPListener>[3]/frmAS2RestPartner[4]?HTTPInterface=syncreply[5]&style=CMA[6]&mode=CMAFormConverter[7]

Example: http://10.0.105.215:18080/cmahttp/frmRestPartner?HTTPInterfac=syncreply&mode=CMAFormConverter

1) IP of BIS instance.
2) Port of HTTP listener.
3) URI of HTTP Controller-Sync application of HTTP listener.
4) Form name.
5) Marker for synchronous HTTP request.
6) Style parameter activating CMA form conversion (the table below shows which parameters and values are available).
7) Trigger matching expression of correct rule.

This sends a synchronous HTTP request calling the CMA form conversion script. Several HTTP query strings are available.
Conversion parameters:

Key	Value	Description
style	CMA	Default value. Conversion of BIS REST form XML into CMA form XML.
	HTML	Generation of CMA form HTML doc.
flat	true	Creates flat CMA form, i.e. all form components on 1 target form tab.
	false	Default value. Creates tabbed CMA form, i.e. each source form tab is converted to a target form tab. Note that moving components between CMA forms tabs is not supported.
cbo2txtbox		Check which mode makes more sense. If a mixture is needed, create both versions and merge the forms manually. Note that CMA forms do not offer editable comboboxes.
	true	Default value. Converts editable GF comboboxes into editable CMA text fields.[1] Should be used in case of combobox lists that contain no real values or values that cannot be predefined such as unique ID (refer to example below).

Key	Value	Description
	false	Converts editable GF comboboxes into non-editable CMA comboboxes.[2] Makes only sense if the combobox list contains a set of predefined values. This set is restricted and inflexible but enables correct values and quick selection.
language	en	Uses English NLS translation. Note that NLS translation in CMA is not supported. All forms need to be populated with translated texts and labels.
	<xx>	Depends on available NLS translation sources.

1) Example for a GF combobox that should be converted into a text field:

```
<combobox id="cobAS2Id" ...>
     <listElement label="" />
     <listElement value="$(ANY_ID)" label="&lt;ALLOW ANY&gt;" translate="false" />
</combobox>
```

2) Example for a GF combobox that can be converted into a combobox (compare the restriction of necessary list elements in a combobox (correct values, quick selection, not flexible) with the possibility to enter all possible values into a text field (error prone/typos, unsupported values):

```
<combobox id="cobContentType" ...>
     <listElement label="text/plain" />
     <listElement label="application/edifact" />
     <listElement label=" application/edi-x12" />
     <listElement label=" application/octet-stream" />
     <listElement label=" application/xml" />
</combobox>
```

Conversion Features and Limitations

Generic form (GF)	CMA form	Comments
Tab	Group	Each group is represented as tab: Number of target tabs depends on flat parameter value. Either 1 tab totally (flag=true) or 1 target tab for each source tab.
Box	Sub-group	• GF boxes can be nested. • CMA sub-groups have only 1 level: GF box nesting is flattened.
Textfield	Editor field TEXT	Default attributes are not converted.
Numeric	Editor field TEXT	Default attributes are not converted.
Password	Editor field TEXT	Default attributes are not converted.
Combobox	Option field CHOOSER	CMA features and restrictions: • Supports fix/predefined value lists oly. Dynamic list loading is not supported. • Not editable. • No max. length attribute. • "Default" attributes are converted into "preselected" attributes.

Generic form (GF)	CMA form	Comments
	Editor field TEXT	Because of restrictions the GF comboboxes can be converted into text fields if they are: • Editable • Dynamically loaded
Listelement	Option	Labels are converted into option labels. Values, if existing, are passed into option IDs. More complex GF listelements are not supported.
Checkbox	Option field CHECKBOX	Default attributes are not converted.
Numeric spinner	Editor field TEXT	Can be changed to NUMERIC type or extended by a regular expression.
Rollover list	Upload field	Not supported by CMA. Only possibility is the upload of a single certificate.
Required flag	Mandatory input type	
Default value	Default value	Only supported by comboboxes.
Field ID	Export name	Boths IDs are expected to be the same: **CMA export names must always match the field IDs of the GF form** in the BIS repository. **Never change the CMA export** name otherwise form initialization and data export will not work properly any more.
Label	Label	Actual labels depend on selected NLS resource. Note that CMA form labels are restricted to a **max length 40**. Labels that are longer are cut.
Tooltip	Description	Actual labels depend on selected NLS resource.
Dependency	Visibility constraint	Most standard dependencies are converted into visibility constraints: Dependencies on combobox list entries cannot be converted, e.g. <dependency condition="cobServiceReceiptMode[1]='true'

##

1.24.6.14.5 How To: CMA Configuration

Tasks:

- Create a template / survey / self service
- Import a CMA form
- Adapt CMA form

- Configure form initialization
- Configure compliance check
- Configure data export

CMA Template and Project Creation

1. Go to *Projects* > *Templates*.
2. Click *New* > *From Scratch*.
3. The *Create New Template* dialog is displayed.
4. Select desired *Type* 'Survey' or 'Self Service'.
5. Enter a meaningful name into *Template*.
6. Click the *OK* button. The *Template Survey* screen is displayed. Current focus is at *Sheets* tab.

Similar for campaigns. If a template exists, the campaign can be created from this template by clicking *New > From Template*.

CMA Form Import

Supported for both templates and campaigns/projects. It is recommended to import a form into a template that can be easily reused by campaigns.

1. Go to *Sheets* tab of template or campaign/project.
2. Click *New* > *From Backup*.
3. Select the (generated) CMA form file. Upon double click, the file is loaded, validated and imported. As long as validation takes place, the background of the File field is colored green. This can take some time.
   If validation succeeds, the green background vanishes.
   If validation fails, an error message is displayed.
4. If validation succeeded, click the *Next* button.
5. Enter the name of the *Sheet*, e.g. 'AS2 Data'.
6. Click the *Finish* button.

CMA Form Adaptation

Adapt the AS2 sheet according to your needs. Decide which changes are to be done in the template (reusability) and which in the actual survey campaign or self service.

1. Go to *Sheets* tab of template or campaign/project.
2. Select and open the sheet.
3. Click the *Modify* button.
4. Adapt the components of the sheet. Refer to General Form Adaption for more details.
5. Click the *Save* button.

Form Initialization

Configurable for both templates and campaigns:

Templates: Go to *Projects* > *Templates* > *Details* > *Properties* tab

Campaigns: Go to *Projects* > *Administration* > *Details* > *Settings* tab

1. Go to *External validation* group

2. Switch on *Activate* check box
3. Select *WebService (SOAP)* or *HTTP* request type
4. Enter URL (see below)
5. Switch on *Use for Initialization* check box
6. Save

WebService URL:
http://10.0.101.102:18080/cma/CMAExternalProcessingPort
A subject is added to the webservice request: 'INIT'. Can be used for evaluation in BIS rules or BIS criteria.

HTTP URL:
http://10.0.101.102:18080/cmahttp/AS2?MODE=<InvocationModeVariable>
Look in the *Use variable* combobox for either 'Mode (INIT, VALIDATION, EXPORT)' or 'Invocation Mode' depending the CMA version.
A query string '?MODE=INIT' is added to the HTTP request that can be evaluated by the receiving BIS events or BIS processes.

Compliance Check

Consists of 2 parts:

- Creation of the Compliance set including the service
- Adding the CMA Format Test

Creation of the Compliance Set

1. Go to *Configuration* > *Compliance Sets*.
2. Click the *New* button.
3. Enter a meaningful name in *Configuration*.
4. Select *Request Type* 'WebService (SOAP)'.
5. Enter the URL: http://10.0.101.102:18080/cma/CMAExternalProcessingPort.
   A subject is added to the webservice request: 'COMPLIANCE_CHECK'. Can be used for evaluation in BIS rules or BIS criteria.
6. Click the *Create* button.
7. A new service is created and display in the overview screen. Go to the details.
8. Go to the *Compliance Sets* tab.
9. Click the *New* button.
10. Enter the data for your compliance set, e.g.
    Configuration := 'INVOIC'
    Validation Type := 'CONTRL'
    Compliance Set := 'compliance-2.1.1-SNAPSHOT'
    Schema := 'D06A/INVOIC'
    Document Type := 'EDIFACT'
11. Click the *Create* button.

Adding the Format Test

1. Go to the project/campaign.
2. Go to the sheet.
3. Enable the edit mode (modify button).
4. Add a new page.
5. Select *Page Type* 'Format test'.

6. Click *Next* and *Finish* buttons.
7. The group properties wizard is displayed. Enter a *Display Name*.
8. Select the *Compliance Set*, e.g. 'INVOIC'.
9. Select the *Verification Level*, e.g. 'Strict (COMPLIANT only)'.
10. Enter the *Number of Successful Tests*. The default value is 3.
11. Click *Next* and *Finish* buttons.
12. The new *Format Test* page is displayed.

Data Export

Configurable for both templates and campaigns:

Templates: Go to *Projects* > *Templates* > *Details* > *Export Rules* tab

Campaigns: Go to *Projects* > *Administration* > *Details* > *Export Rules* tab

1. Click the *New* button
2. The wizard is displayed. Enter the name of the rule, e.g. 'Export AS2'
3. Click the *Next* button.
4. A condition is not needed. Click the *Next* button again.
5. Enter the preferred *Type of Export*: 'HTTP Export' or 'WebService CMAExternalProcessing'
6. Click the *Next* button again.
7. Enter *URL* (see below).
8. In case of an HTTP export:
   HTTP Method := 'POST'
   Export Format := 'Project schema'
   Type of the Response := 'text/xml'
9. Click the *Finish* button.

WebService URL:
http://10.0.101.102:18080/cma/CMAExternalProcessingPort
A subject is added to the webservice request: 'EXPORT'. Can be used for rules or criteria.

HTTP URL:
http://10.0.101.102:18080/cmahttp/AS2?MODE=EXPORT
A query string '?MODE=EXPORT' is added to the HTTP request that can be evaluated by the receiving BIS events or BIS processes.

1.24.6.14.6 How To: BIS Configuration of CMA Surveys

2 CMA request types possible: HTTP (POST) and WebService (SOAP).

Requirements of installation:

- Web Service Controller
- HTTP Client and Controller
- BIC Content Extractor (in case of WebService requests, needed by MFT-CMAExternalProcessing event)
- AS2 REST GUI and service
- Keystore Management Client
- MTP tool
- Incoming change requests GUI

Sample Configuration WebService Initial Population

Requirements:

- WebServices Controller adapter started
- BIC Content Extractor Node adapter started
- HTTP Client and HTTP Controller adapters started
- Script Executor adapter started

- HTTP server partner referencing CMA installation (IP)
- HTTP listener with application 'Web Services Controller' and URI '/cma/*'
- Rule for interface 'CMAExternalProcessingPort' and operation 'CMAExternalProcessing', name: 'cma:INIT', condition: '//subject='INIT'

1.	Criteria	From adapter	'Webservice'	Internal value 'WEBSRV'
		From name	CMA project name	CMA project name, extracted from
		From ID	'*'	Email address of CMA user
		From subject	'INIT'	Fix value, the CMA request type.
		Doc type	'XML'	Fix value, process attachment is always XML.
2.	acBookmark	Bookmark	'CMA_REQUEST'	Bookmark the received process attachment. Will be needed in step 4 by acCMAInit.
3.	acGetBISMasterData	Template dataset ID	'/api/as2/NEU/NEUPARTNER'	
		Queue	'direct:	
4.	acCMAInit	CMA init request bookmark	'CMA_REQUEST'	The received process attachment bookmarked in step 2
		Queue	'direct:'	

#

Sample Configuration WebService Data Export

Requirements:

- WebServices Controller adapter started
- BIC Content Extractor Node adapter started
- HTTP Client and HTTP Controller adapters started
- Script Executor adapter started

- HTTP server partner referencing CMA installation (IP)

- HTTP listener with application 'Web Services Controller' and URI '/cma/*'
- Rule for interface 'CMAExternalProcessingPort' and operation 'CMAExternalProcessing', name 'cma:EXPORT', condition: '//subject='EXPORT'

1.	Criteria	From adapter	'Webservice'	
		From name	CMA project name	CMA project name
		From subject	'INIT'	Fix value, the CMA request type.
		Doc type	'XML'	Fix value, process attachment is always XML.
2.	acBookmark	Bookmark	'CMA_REQUEST'	Bookmark the received process attachment. Will be needed in step 4 by acCMA2ChangeRequest.
3.	acGetBISMasterData	Template dataset ID	'/api/as2/NEU/NEUPARTNER'	
		Queue	'direct:	
4.	acCMA2ChangeRequest	Change request mode	'Survey'	The mode to be used for surveys. Allows data creation only and fails when trying to update existing data.
		CMA init request bookmark	'CMA_REQUEST'	The received process attachment bookmarked in step 2
		Embed certificates in CR	'False'	
		Queue	'direct:'	
5.	acSetBISMasterData	BIS API service	'/api/mtp'	Access to change request service
		Queue	'direct:'	

#

Sample Configuration HTTP Initial Population

Requirements:

- HTTP Client and HTTP Controller adapters started
- Script Executor adapter started

- HTTP server partner referencing CMA installation (IP)
- HTTP listener with application 'HTTP Controller-Sync' and URI '/cmahttp/*'
- Rule for interface 'SHC' and operation 'initiateSync', name: 'cma:SHC_SYNC', condition: '1=1'

1.	Criteria	From adapter	'HTTP synchronous (Server)'	Synchronous HTTP Server call from CMA
		From address	'<any address>'	HTTP address from HTTP Server Partner
		From subject	'MODE=INIT*'[2]	Fix value, the CMA request type
		From path	'/cmahttp/AS2*'	HTTP listener path
		Doc type	'XML'	Fix value, process attachment is always XML
2.	acBookmark[1]			*Refer to WebService version above*
3.	acGetBISMasterData[1]			*Refer to WebService version above*
4.	acCMAInit[1]			*Refer to WebService version above*
5.	acHTTPSyncReply	Response type	'Attachment'	Returns the process attachment created by acCMAInit to CMA.
		Content type	'text/xml'	
		Status code	'200 OK'	

1. Activities are configured the same way as above in the WebService section.
2. Not set automatically by CMA, needs to be entered manually in Projects > Administration > Project Details View > Settings Tab > External Validation > URL. For instance: http://10.0.101.101:18080/cmahttp/AS2?MODE=.

Sample Configuration HTTP Data Export

Requirements:

- HTTP Client and HTTP Controller adapters started
- Script Executor adapter started

- HTTP server partner referencing CMA installation (IP)
- HTTP listener with application 'HTTP Controller-Sync' and URI '/cmahttp/*'
- Rule for interface 'SHC' and operation 'initiateSync', name: 'cma:SHC_SYNC', condition: '1=1'

1.	Criteria	From adapter	'HTTP synchronous (Server)'	Synchronous HTTP Server call from CMA
		From address	'<any address>'	HTTP address from HTTP Server Partner

		From subject	'MODE=INIT*'[2]	Fix value, the CMA request type
		From path	'/cmahttp/AS2*'	
		Doc type	'XML'	Fix value, process attachment is always XML
2.	acBookmark[1]			*Refer to WebService version above*
3.	acSetProperties	Key value pair 1	'CMA_SELFSERVICE_MODE=EXPORT'	
4.	acGetBISMasterData[1]			*Refer to WebService version above*
5.	acCMA2ChangeRequest[1]			*Refer to WebService version above*
6.	acSetBISMasterData[1]			*Refer to WebService version above*
7.	acHTTPSyncReply	Response type	'Attachment'	Returns the process attachment created by acCMA2ChangeRequest to CMA.
		Content type	'text/xml'	
		Status code	'200 OK'	

[1] Activities are configured the same way as above in the WebService section.
[2] Not set automatically by CMA, needs to be entered manually in Projects > Administration > Project Details View > Settings Tab > External Validation > URL. For instance: http://10.0.101.101:18080/cmahttp/AS2?MODE=EXPORT.

1.24.6.14.7 How To: BIS Configuration of CMA Self Services

2 CMA request types possible: HTTP (POST) and WebService (SOAP).

Requirements of installation:

- Web Service Controller
- HTTP Client and Controller
- BIC Content Extractor (in case of WebService requests, needed by MFT-CMAExternalProcessing event)
- AS2 REST GUI and service
- Keystore Management Client
- MTP tool
- Incoming change requests GUI

Sample Configuration WebService Initial Population

Requirements:

- WebServices Controller adapter started
- BIC Content Extractor Node adapter started
- HTTP Client and HTTP Controller adapters started
- Script Executor adapter started

- HTTP server partner referencing CMA installation (IP)
- HTTP listener with application 'Web Services Controller' and URI '/cma/*'
- Rule for interface 'CMAExternalProcessingPort' and operation 'CMAExternalProcessing', name: 'cma:INIT', condition: '//subject='INIT'

1.	Criteria	From adapter	'Webservice'	
		From name	CMA project name	CMA project name
		From subject	'INIT'	Fix value, the CMA request type.
		Doc type	'XML'	Fix value, process attachment is always XML.
2.	acBookmark	Bookmark	'CMA_REQUEST'	Bookmark the received process attachment. Will be needed in step 4 by acCMAInit.
3.	acGetBISMasterDataForSelfService	TPM share point	'<use default>'	
		Template dataset ID	'/api/as2/NEU/NEUPARTNER'	
		Queue	'direct:	
4.	acCMAInit	CMA init request bookmark	'CMA_REQUEST'	The received process attachment bookmarked in step 2
		Queue	'direct:'	

#

Sample Configuration WebService Data Export

Requirements:

- WebServices Controller adapter started
- BIC Content Extractor Node adapter started
- HTTP Client and HTTP Controller adapters started
- Script Executor adapter started

- HTTP server partner referencing CMA installation (IP)
- HTTP listener with application 'Web Services Controller' and URI '/cma/*'
- Rule for interface 'CMAExternalProcessingPort' and operation 'CMAExternalProcessing', name 'cma:EXPORT', condition: '//subject='EXPORT'

1.	Criteria	From adapter	'Webservice'	

		From name	CMA project name	CMA project name
		From subject	'INIT'	Fix value, the CMA request type.
		Doc type	'XML'	Fix value, process attachment is always XML.
2.	acBookmark	Bookmark	'CMA_REQUEST'	Bookmark the received process attachment. Will be needed in step 4 by acCMA2ChangeRequest.
3.	acGetBISMasterDataForSelfService	TPM share point	'<use default>'	Use the automatically created default share point 'CMAExport' as TPM root node.
		Template dataset ID	'/api/as2/NEU/NEUPARTNER'	
		Queue	'direct:	
4.	acCMA2ChangeRequest	Change request mode	'Survey'	The mode to be used for surveys. Allows data creation only and fails when trying to update existing data.
		CMA init request bookmark	'CMA_REQUEST'	The received process attachment bookmarked in step 2
		Embed certificates in CR	'False'	
		Queue	'direct:'	
5.	acSetBISMasterData	BIS API service	'/api/mtp'	Access to change request service
		Queue	'direct:'	

#

Sample Configuration HTTP Initial Population

#

Sample Configuration HTTP Data Export

Requirements:

- WebServices Controller adapter started
- BIC Content Extractor Node adapter started
- HTTP Client and HTTP Controller adapters started
- Script Executor adapter started

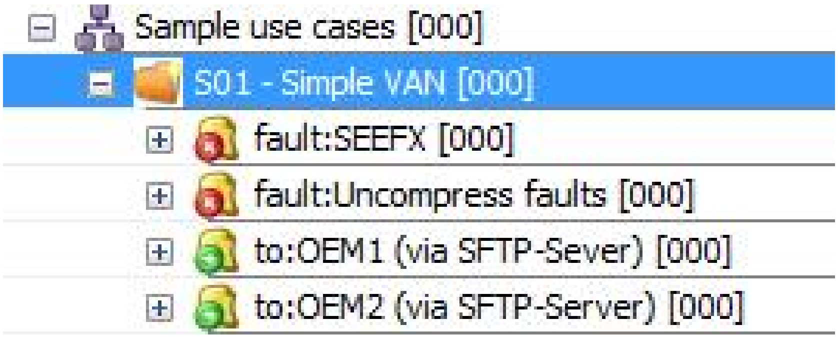
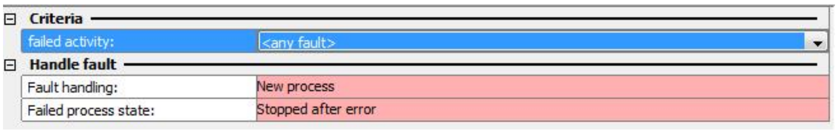
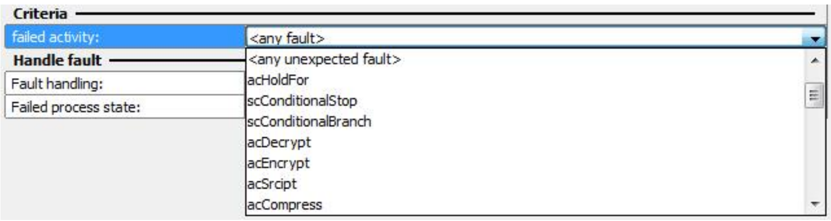
- HTTP server partner referencing CMA installation (IP)
- HTTP listener with application 'Web Services Controller' and URI '/cma/*'
- Rule for interface 'CMAExternalProcessingPort' and operation 'CMAExternalProcessing', name 'cma:EXPORT', condition: '//subject='EXPORT'

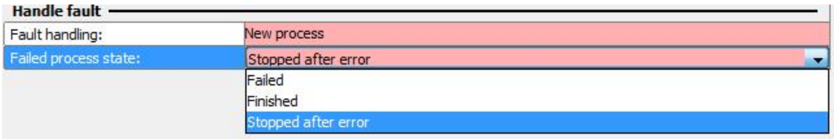
1.	Criteria	From adapter	'Webservice'	
		From name	CMA project name	CMA project name
		From subject	'INIT'	Fix value, the CMA request type.
		Doc type	'XML'	Fix value, process attachment is always XML.
2.	acBookmark	Bookmark	'CMA_REQUEST'	Bookmark the received process attachment. Will be needed in step 4 by acCMA2ChangeRequest.
3.	acGetBISMasterDataForSelfService	TPM share point	'<use default>'	Use the automatically created default share point 'CMAExport' as TPM root node.
		Template dataset ID	'/api/as2/NEU/NEUPARTNER'	
		Queue	'direct:	
4.	acCMA2ChangeRequest	Change request mode	'Survey'	The mode to be used for surveys. Allows data creation only and fails when trying to update existing data.
		CMA init request bookmark	'CMA_REQUEST'	The received process attachment bookmarked in step 2
		Embed certificates in CR	'False'	
		Queue	'direct:'	
5.	acSetBISMasterData	BIS API service	'/api/mtp'	Access to change request service
		Queue	'direct:'	

#

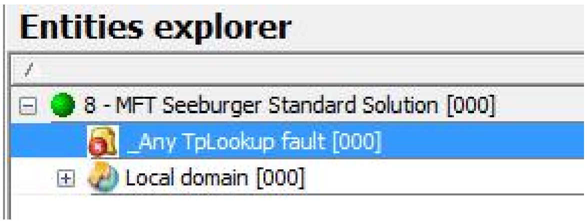
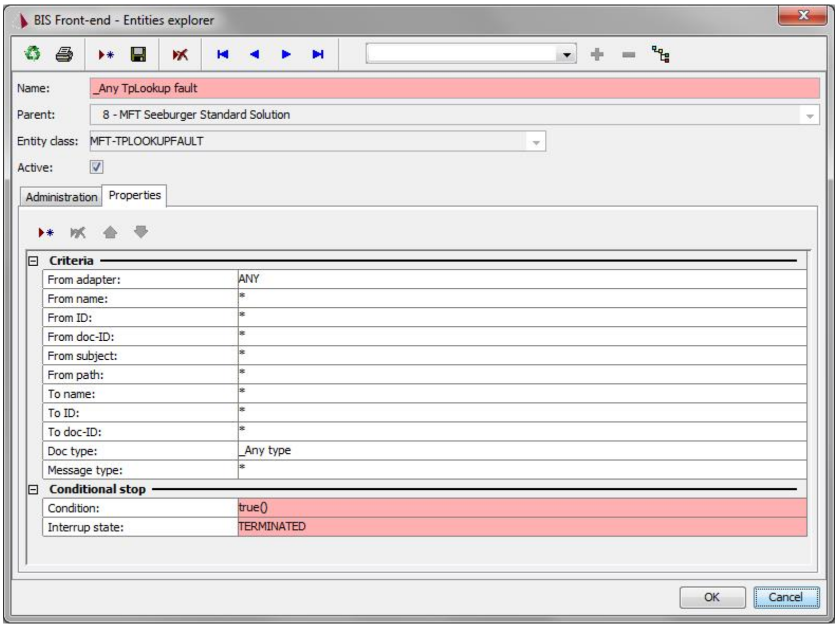
2 Fault Handling

2.1 Configuration

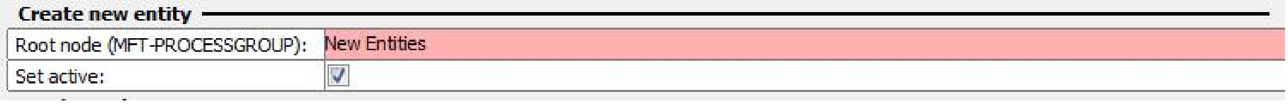
Instruction	Illustration
1. Fault handling can be configured below MFT_PROCESSGROUP entity.	 Sample use cases [000] S01 - Simple VAN [000] fault:SEEFX [000] fault:Uncompress faults [000] to:OEM1 (via SFTP-Sever) [000] to:OEM2 (via SFTP-Server) [000]
2. The minimum configuration consists of a criterion which fault to catch and a handling directive how the failed process shall resume.	 Criteria failed activity: <any fault> Handle fault Fault handling: New process Failed process state: Stopped after error
3. The fault handler can have multiple criterions. A criterion is either the name of the failed activity, e.g. acUnCompress, in case uncompress activity failed or a class of faults like <any failed activity>. • <any fault> : catches any fault • <any failed activity> : catches only failed activity • <any policy violation> : catches policy violation faults • <any unexpected fault> : catches faults which are neither activity faults nor policy violation faults	 Criteria failed activity: <any fault> Handle fault Fault handling: Failed process state: <any unexpected fault> acHoldFor scConditionalStop scConditionalBranch acDecrypt acEncrypt acSrcipt acCompress

Instruction	Illustration
4. After catching a fault the handling of the fault must be specified. Currently only “new process” is supported, what means that the fault process will be executed as new process. The fault process will get the TRACK_ID of the failed process with appendix .err. „Failed process state“ allows to set the state of the failed process. Possible values are: • Failed : the status of the failed process will be set to ERROR. The process cannot be continued • Finished : the status of the failed process will be set to SUCCESS. The process cannot be continued • Stopped after error : the failed process will be interrupted and can be continued A list of activities can be added to the fault process. The attachment processed is the log file of the failed process with the fault description. If a fault handler process fails no further fault handler is called. The fault handler process is stopped with error.	 Handle fault Fault handling: New process Failed process state: Stopped after error

2.2 Catching TP-Lookup Faults (No Entity Found Exception)

Instruction	Illustration
1. With the fault handler described above, it is not possible to catch TP-Lookup faults. For handling TP-Lookup faults, an own process of type MFT-TPLOOKUPFAULT in TPM shall be created. This process can only be created on the same level as the MFT-DOMAIN object, because it is common for all.	 Entities explorer / ⊟ 8 - MFT Seeburger Standard Solution [000] _Any TpLookup fault [000] ⊞ Local domain [000]
2. This is a fall back configuration for all TP-Lookups. You should configure a general handling for TPLookup faults here.	 BIS Front-end - Entities explorer Name: _Any TpLookup fault Parent: 8 - MFT Seeburger Standard Solution Entity class: MFT-TPLOOKUPFAULT Active: ☑ Administration | Properties ⊟ Criteria From adapter: ANY From name: * From ID: * From doc-ID: * From subject: * From path: * To name: * To ID: * To doc-ID: * Doc type: _Any type Message type: * ⊟ Conditional stop Condition: true() Interrup state: TERMINATED OK | Cancel

Auto-Creation of Entities



Create new entity

Root node (MFT-PROCESSGROUP):	New Entities
Set active:	☑

MFT-TPLOOKUPFAULT entities can be configured to create a new MFT-PROCESS entity in BIS trading partner management. The root node points to an existing MFT-PROCESSGROUP entity below, which the new entity is created.

The new entity is created with an exact matching CriteriaMFTType. It does not contain any activities.
The name of the new created entity starts with exclamation mark followed with values created from FROM_NAME/FROM_PATH properties, followed by the track id of the process which created this entity.

The “set active” switch activates the new created entity. As a result, other incoming files which match this entity are processed with success.

If the “Set active” switch is off, files which would match this entity will created a new identical identity.

Important: To avoid unexpected growth of the number of added entities, the maximum number of child entities below the parent entity is limited to 100. If this number is exceeded, the process stops with an error and the additional configured process flow is not executed.

Recommended usage: MFT-TPLOOKUPFAULT entity should be used to create a new entity and forward an email to an operator about the fact that a new entity was created. Use property ENTITY_CAPTION to inform about the name of the new created entity. The status of the process should be set to ERROR. This makes it easier to find the process. The operator should move and rename the automatic created entity.

3 Property References

3.1 Notation of Properties, System Variables and Expressions

${} – References a system variable. System vairables are global, because you specify them in the GUI for all processes.

$[] – References a property. Properties are local, their values are available for the currently running process instance only. They are located in a special process variable called property bag. Only properties added to such a property bag are accessible in the process. In the MFT solution, all activities have a property bag.
New properties can be added to the property bag, for example with a mapping. All output values specified in the mapping are returned to the process and added to its property bag. Thus, the properties in the porperty bag are available for all subsequent activities.

$() – Specifies an XPath expression. For instance, COUNTER is defined as a system variable which expands to an expression $(seeutil:decimalFormat(seeutil:nextCounter('COUNTER'), '0000000000')). An overview of available XPath functions can be found in the SEEBURGER XPath Functions book (link).

3.2 Variable Reference

3.2.1 System Variables

General MFT System Variables

System Variable	Default Value	Description
MFT-PROCESSOR-DESTINATION	direct:	Because logging is made via a mapping this is the queue
MFT-REGISTER-TRIGGER-DIRECT	TRUE	Currently only TRUE is supported
MFT-TPM-ONLY-ONE-MATCH-ALLOWED	FALSE	If set to TRUE lookup CriteriaMFT will fail if more than one entity matches the criterion.
MFT-TRACKID-COUNTER	$(concat(seeutil:getEnv(), seeutil:decimalFormat(seeutil:nextCounter ('MFT_TRACKING_COUNTER'), '0000000')))	The XPath expression which creates the process TRACKID.

Message Tracking Related System Variables

System Variable	Default Value	Description
SSC-MT-ROOT-PATH	../../../../MT_ROOT/	Message tracking root folder – this is the correct path if message tracking is installed together with BIS6 below the same root folder.
MFT-LOG-SUB-PATH	/mft.log/	Sub-path for log files below SSC-MT-ROOT-PATH
MFT-MT-DATA-SUB-PATH		A predefined sub path below SSC-MT-ROOT-PATH for storing payload data. The activity uses this path.
MFT-MTLOGGER-DESTINATION	direct:	sync: or direct: defines the bic queue for message tracking
MFT-MTLOGGER-MAPPING	CC_MT_Mapping	Mapping name used for message tracking
MFT-MT-LOG-NOTIFICATIONS	TRUE	If TRUE, a MT entry is written in case the component replies with status SUCCESS_NOTIFY (see activity development framework)

Activity Specific System Variables

System Variable	Default Value	Description
MFT-FILELISTCREATOR	MFT-FILELISTCREATOR	The mapping which is used to create the standard file list used in .
MFT-AS2-MDN-HANDLING	AFTER_RECEIVE, GATEWAY_MODE	In case AS2 sender requested an asynchronous MDN: AFTER_RECEIVE – the MDN is sent immediately after process started (always positive MDN). GATEWAY_MODE – the MDN is sent after the process is finished. (in case process failed and a fault handling was configured to resumes the process a negative MDN can be sent).
MFT-EUN-EXTERNAL-PROGRAM-SCRIPT	CallProgram.bsh	The beanshell script which is used in acRunExternalProgramm component

#

3.2.2 Global Process Properties

These properties are created from the initiating event and remain the same during process execution.

- Part 1: File related adapters (hotfolder, file system, FTP, SFTP)
- Part 2: Several DT adapters (AS2, AS4, JMS, MAIL, OFTP, X400)
- Part 3: HTTP and SAP adapters
- Part 4: Various remaining adapter types (SLMP, SEEFX, WebService, Time Event, Database Event)

File Related Adapters

All mainly file related adapters with similar properties.

	Hotfolder (classic)	**FileSystem (Server and client)**	**FTP (Client)**	**FTP (Server)**	**SFTP (Client)**	**SFTP (Server)**
FROM_ADAPTER	HOTFLD	FILSYS	FTPCLI	FTPSRV	SFTCLI	SFTSVR
FROM_NAME	Name of folder listener i.e. groupname of hofolder event data	BIS address name of file request or 'NOT_SET'	BIS origin address name or 'NOT_SET'	BIS origin address name	BIS origin address name	BIS origin address name or 'NOT_SET'
FROM_UUID	UUID of folder listener i.e. groupID of hotfolder eventData	BIS address UUID of file request or 'NOT_SET'	BIS origin address UUID or UUID placeholder[1]	BIS origin address UUID	BIS origin address UUID or UUID placeholder[1]	BIS origin address UUID
FROM_ID	Same as FROM_NAME	unused	User name of FTP request partnerSettings (configured in FTP address)	User name of FTP request partnerSettings (configured in FTP address)	User name of SFTP request partnerSettings (configured in SFTP address)	User name of SFTP request partnerSettings (configured in SFTP address)
FROM_PATH	File name of hotfolder eventData	File name of file request file parameter	Remote file name of FTP request without path	File name of FTP request	Remote file name of SFTP request	File name of SFTP request
FROM_SUBJECT	File path without file name ending with /	Full path of file request file parameter, ending with a path separator '/'	Sub-path of user home folder ending with / ex. / USER1/ (may be empty)	Sub-path of user home folder ending with / ex. / USER1/ (may be empty)	Sub-path of user home folder ending with / ex. / USER1/ (may be empty)	Sub-path of user home folder ending with / ex. / USER1/ (may be empty)
TO_NAME	hotfolderName of hotfolder eventData	unused	unused	unused	unused	unused
TO_UUID	hotfolderID of hotfolder eventData	unused	unused	unused	unused	unused
TO_ID	unused	unused	unused	unused	unused	unused

1) UUID placeholder '????????-????- ????-????- ????????????'.

DT adapters

Several DT adapters except HTTP, FTP and SFTP.

	AS2 (Server)	AS4 (Server)	JMS	MAIL (Client)	MAIL (Single Flow)	OFTP TCP/IP	X400 / P7
FROM_ADAPTER	AS2SRV	AS4SRV	JMSCLI	MAILCL	MAILSF	OFTSRV	X400P7
FROM_NAME	AS2 request: origin address name	AS4 message senderInformation: address name	JMS message: destination address name	MAIL message: origin address name or 'NOT_SET'	MAIL message: origin address name or 'NOT_SET'	OFTP request: origin address name	P7 request: origin address name
FROM_UUID	AS2 request: origin address ID	AS2 message senderInformation: address ID	JMS message: destination address ID	MAIL message: origin address ID or UUID placeholder[1]	MAIL message: origin address ID	OFTP request: origin address ID	P7 request: origin address ID
FROM_ID	AS2 request: originator AS2 ID	AS4 message senderInformation: party ID	*Identical to FROM_NAME*	MAIL message: original address email	MAIL message: original address email	OFT request: sfidCode (originator) or 'NOT_SET'	P7 request originatorAddress: surName or 'NOT_SET', '/' and givenName appended if givenName is available
FROM_PATH	AS2 request: file name or 'NOT_SET'	'NOT_SET'	JMS message: provider queue	MAIL message: payload name or empty	MAIL message: payload name or 'NOT_SET'	OFTP request: dataSetName (transferInfo) or 'NOT_SET'	P7 request omsCdifInformation: fileName or comment, fileName/ comment if available, 'NOT_SET' if not available
FROM_SUBJECT	AS2 request subject	AS4 message: palyoad file name or 'NOT_SET'	Specified by system variable[2] (see examples below[3])	MAIL message: subject	Mail message: subject	OFTP request: subject or 'NOT_SET'	P7 request: subject or 'NOT_SET'
TO_NAME	BIS destination address name	AS4 receiver address name	#	BIS destination address name	Destination address name	BIS destinator address name	BIS own address name
TO_UUID	BIS destination address UUID	AS4 receiver UUID	unused	BIS destination address UUID	Destination address UUID	BIS destinator address UUID	BIS own address UUID
TO_ID	AS2 ID from AS2 request recipient	AS4 receiver party ID	#	Destination email address: mailbox used from received mail message mailboxOwner settings	Mailbox user	SFID code from OFTP request destinator	P7 request: destination address, 'sub name/ given name' if available, 'NOT_SET' if none is available

1) UUID placeholder '????????-????- ????-????- ????????????'.
2) System variable ${MFT-SUBJECT-PATTERN-JMS}.
3) Examples: "$[GROUP_ID]" or "$[MESSAGE_ID]" or "${JMS_REPLY_ DESTINATION}"

HTTP and SAP adapters

HTTP and SAP client adapters as well as synchronous and asynchronous server adapters.

	HTTP (Client)	SHC-Sync. (HTTP Server)	SHC-Async. (HTTP Server)	IDOC	SAP RFC (Async.)	SAP RFX (Sync.)
FROM_ADAPTER	HTTPCL	SHCSYN	SHCASY	IDOCTL	RFCCTL	RFCSYN
FROM_NAME	HTTP request: destination address name	SHC request: destination address name	SHC request: destination address name	IDOC event: registration name	RFC request: SAP client or 'NOT_SET'	RFC request: SAP client or 'NOT_SET'
FROM_UUID	HTTP request: destination address UUID	SHC request: destination address UUID	SHC request: destination address UUID	IDOC event: registration ID	RFC request: address ID	RFC request: address ID
FROM_ID	HTTP request: user (clientInfo) or 'NOT_SET'	SHC request: remote host (serverInfo)	SHC request: remote host (serverInfo)	*Identical to FROM_NAME*	RFC request: connection ID	RFC request: connection ID
FROM_PATH	'NOT_SET'	SHC request: requestURI or 'NOT_SET'	SHC request: requestURI	IDOC event: program ID	RFC request: function name or 'NOT_SET'	RFC request: function name or 'NOT_SET'
FROM_SUBJECT	HTTP request: subject or 'NOT_SET'	SHC request: queryString or 'NOT_SET'	SHC request: queryString or 'NOT_SET'	Specified by system variable[1], semicolon separated list (see example below[2])	RFC request: program ID or 'NOT_SET'	RFC request: program ID or 'NOT_SET'
TO_NAME	unused	unused	unused	'NOT_SET'	unused	unused
TO_UUID	unused	unused	unused	'NOT_SET'	unused	unused
TO_ID	unused	unused	unused	'NOT_SET'	unused	unused

1) System variable ${MFT_SUBJECT_ PATTERN_IDOC}.
2) Example for semicolon separated list: 'MESCOD=$[SAP_MESCOD];MESFCT=$[SAP_MESFFCT];...". The properties inside are resolved by extracted IDOC metadata.

Various adapter types

Collection of the remaining adapter types:

	SLMP SEE LINK	SEE FX	BIS FX	Upload File WebServic	Generic Web Service	Splitter	Time Event (Scheduler	Database Event
FROM_ADAPTER	SLMPCL	SEE-FX	BIS-FX	UPLOAD	WEBSRV	SPLFIL	TIMEVT	DBEVNT
FROM_NAME	SLMP message: address name (originator)	Name of creator of file (createdBy of download file response FileInfo)	SHC request: destination address name or empty	fromName (see wsdl)	Decider rule name	fromName	PARAM1 (in scheduler config)	JDBC request DBListenerEvent: port name
FROM_UUID	SLMP message: address ID (originator)	Not set	SHC request: destination address id	UUID placeholder[1]	UUID placeholder[1]	fromUUID	unused	JDBC request DBListenerEvent: port ID or empty
FROM_ID	SLMPmessage: SLMPID (originator)	Not set	SHC request: query string for 'operator' or 'NOT_SET'	Id (see wsdl)	Port type	fromID	unused	JDBCD request DBListenerEvent: engine ID or 'NOT_SET'
FROM_PATH	SLMP message: payload file name or 'NOT_SET'	File name (from download file response FileInfo)	SHC request: query string for 'path' or 'NOT_SET'	Path (see wsdl)	Address (EndPoint Reference)	fromPath	PARAM2 (in scheduler config)	JDBC request DBListenerEvent: data source name or 'NOT_SET'
FROM_SUBJECT	SLMP message: subject	SEE FX path without file name ending with /	SHC request: query string for 'operation' or 'ReceiveUpload'	Subject (see wsdl)	Operation	fromSubject	Name of scheduler task (in scheduler config)	JDBC request DBListenerEvent: table name or 'NOT_SET'
TO_NAME	BIS address name from SLMP request recipient	folder of event or 'My Files'	SHC request: query string for 'sysuser' or empty	Can be set via property TO_NAME (see wsdl)	unused	unused	PARAM3 (in scheduler config)	'NOT_SET'
TO_UUID	BIS address UUID from SLMP request recipient	unused	'NOT_SET'	unused	unused	unused	unused	'NOT_SET'
TO_ID	SLMPID from SLMP request recipient	unused	SHC request: query string for 'e-mail' or empty	Can be set via property TO_ID (see wsdl)	unused	unused	unused	'NOT_SET'

1) UUID placeholder 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'.

3.2.2.1 Additional properties in case of qualified documents (BIC Classifier)

These fields are available if the BIC classifier could recognize the document. Depending on DOC_TYPE, additional fields for sender and receiver are filled.

Property name	Meaning
DOC_TYPE	The type of the document, e.g. ANSI.X12, EDIFACT, PDF, etc.
MESSAGE_TYPE	The message type of the first message in the document
FROM_DOCID	The sender ID from the document, e.g. ILN
TO_DOCID	The receiver ID from the document

#

3.2.2.2 Additional Properties used in the Process

Additional process related properties:

Property name

	Meaning
TRACK_ID	The track ID of the process, shown in the process monitor and used in message tracking
ORIGINAL_FILE_ID	The attachment UUID of the original file

#

3.2.2.3 Additional Properties created by scForEach

Properties set during ForEach execution

The *scForEach* activity sets the following properties while looping the CSV entries in case of FILELIST type:

Property name[1]	Existing	File activity meaning[2]	JMS activity meaning[3]
ENTRY_FILE_NAME	Always	File name	JMS message ID
ENTRY_FILE_PATH	Always	File path (may be empty)	Not available: value is 'NOT_SET'
ENTRY_FILE_SIZE	Always	File size	Not available: value is 'NOT_SET'
ENTRY_FILE_DATE	Always	File date (may be empty)	Message date
ENTRY_FILE_SUB_PATH	Not always	Sub path	Not available

1) The table shows the meanings for the properties depending on the activity that created the FILELIST.
2) File activities are the following: *acFileClientList*, *acFTPList*, *acSFTPList*, *acMimeList*, *acZIPList*, *acSEEFXInboxList*, *acSEEFXViewList2*. Each CSV entry represents a file.
3) JMS activities are the following: *acJMSListQueue*. Each CSV entry represents a message.

The *scForEach* activity sets the following properties while looping the CSV entries in case of ADDRESSLIST type:

Property name[1]	**Existing**	**Meaning**[2]
ENTRY_ADDRESS_NAME	Always	Address name
ENTRY_ADDRESS_UUID	Always	Address UUID
ENTRY_CONNECTION_UUID	Not always	Connection UUID
ENTRY_PROTOCOL	Always	DT protocol, either 'SFTP' or 'SLMP'

1) The table shows the meanings for the properties depending on the activity that created the ADDRESSLIST.
2) Address activities are the following: *acAddressList*. Each CSV entry represents a master data address.

Properties set after scForEach execution

Property name	**Meaning**
LIST_TYPE	FILELIST or ADDRESSLIST
FOREACH_COUNT	Number of loops
FOREACH_STATUS	• 200 success • 204 in case errors have occurred during at least one of the ForEach executions • 205 success, empty list • 415 invalid list type • 500 no sub entity available

#

3.3 Using Properties in Processes

There is a set of reliable variables, but during the process, depending on the components, more variables can be added.

The easiest way to inspect the variables available before executing a certain operation is to stop the process with the acConditionalStop activity, and set the status to ERROR.

Then right-click on the process listed in inspector overview and from the choice, select “attachements” from the appearing list. Open the log file attached to pickupLogFile activity.

The log file shows a dump of the current properties and their values.

Example:

```
key="TO_UUID"   value="NOT_SET"
key="TO_NAME"   value="/My Files/TO SEEBURGER VAN/SUPL2"
key="TO_ID"  value="NOT_SET"
key="TO_DOCID"  value="6038934473"
key="FROM_DOCID"  value="6111470100"
key="MSG_TYPE"  value="ANSIX12_TX"
key="TRANSACTION_ID"   value="0000000757"
key="ORIGINAL_FILE_ID"  value="c64e8730-fb31-11e1-8881-17050a00672c"
key="POLICY_ZONE"  value="INTRANET"
key="LOG_PATH"  value="../../../../MT_ROOT//mft.log//0000000757.log"
key="TRACKID"    value="0000000757"
```

```
key="FROM_ADAPTER"   value="SEE-FX"
key="FILESIZE"  value="17538"
key="FROM_NAME"  value="VAN1"
key="FROM_UUID"  value="xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
key="FROM_ID"   value="VAN1"
key="DOCTYPE"   value="AnsiX12"
key="FROM_PATH"  value="0000000060.txt"
key="FROM_SUBJECT"  value="/My Files/TO SEEBURGER VAN/SUPL2/"
key="ACTIVITY"  value="Activity[0]"
key="ACTIVITY_NAME"  value="ConditionalStop"
key="STEP_COUNT"  value="0"
key="LOG_CODE"  value="ACTIVITY_FAULT"
key="RESULT"  value="ERROR"
key="STATUS"  value="ERROR"
key="HISTORY"   value="#"
```

4 Message Tracking

4.1 Overview

tbismt_workflow	Example	Internal Description	BIC Mapping	BIS Value	MT Frontend Description
CCLIENTID	1	constant Value			
CWFID	50	Track ID	strTrackID$	TRACK_ID	
CPARENTWFID	<null>				
CMESSAGEID	27ef0380-11e3-11e2-a382-2d650a006772		strFileID$	FILE_ID	
CPARTNER	mft:Testport_Audi		strSndName$	SND_NAME	
CDIRECTION	IN		INITIAL=IN, PENDING=OUT		
CMESSAGETYPE	Inhouse		strFileType$	FILE_TYPE	
CTESTFLAG	1				
CMESSAGESIZE	14		strFileSize$	FILE_SIZE	
CMESSAGEDATETIME	1,349,767,817,000		strTimeStamp$	TIMESTAMP	
CPROCESSINGTIME	1,349,767,834,000		strTimeStamp$	TIMESTAMP	
CPROCESSINGTIMESTART	1,349,767,817,000		strTimeStamp$	TIMESTAMP	
CPROCESSINGTIMEEND	1,349,767,834,000		strTimeStamp$	TIMESTAMP	
CSTATUS	SUCCESS		strStatus$, but Constant Values in Mapping	STATUS	
CCLIENT	<null>				
COBJECTID	<null>				
CFREE1			strToTPMCaption$	TO_TPM_CAPTION	
CFREE2	HOTFLD		strSndAdapter$	SND_ADAPTER	

tbismt_workflow	Example	Internal Description	BIC Mapping	BIS Value	MT Frontend Description
CFREE3	b23ebaa1-fcd3-11e1-ae78-97120a006772		strSndUUID$	SND_UUID	
CFREE4	mft:Testport_Audi		strSndID$	SND_ID	
CFREE5	d:/testdaten/testport_audi/		strSndSubject$	SND_SUBJECT	
CFREE6	testfile_from_audi.txt		strSndPath$	SND_PATH	
CFREE7	ForwardingSEEFX		strActivity$	ACTIVITY	
CFREE8	<null>		strToAdapter$	TO_ADAPTER	
CFREE9	<null>				
CFREE10	<null>				
CFREE11	<null>				
CFREE12	<null>				
CFREE13	<null>		strRestarted$	RESTARTED	
CFREE14	27e78970-11e3-11e2-a382-2d650a006772.p1349767810699		strContextID$	CONTEXT_ID	
CFREE15	<null>		strExecTime$	EXEC_TIME	
CFREE16	<null>				
CFREE17	<null>				
CFREE18	<null>		strErrCode$	ERROR_CODE	
CFREE19	<null>		strErrMsg$	ERROR_MESSAGE	
CFREE20	<null>				
CFREEDECIMAL1	<null>				
CFREEDECIMAL2	<null>				
CFREEDECIMAL3	<null>				
CFREEDECIMAL4	<null>				
CFREEDECIMAL5	<null>				
CSENT	0				
CLINK1	attachment:///mft.log//0000000050.log		attachment:// & strLogFilePath$	LOG_FILE_PATH	
CLINK2	<null>				
CLINK3	<null>				
CLEVEL	0				

tbismt_workflow	Example	Internal Description	BIC Mapping	BIS Value	MT Frontend Description
CROOTWFID	50				
CCHILDRENFLAG	0				
CEXPECTEDCHILDREN	0				
CSTARTOFTIMEINTERVAL	1,349,767,824,344				
CENDOFTIMEINTERVAL	<null>				
CREFDATE	1,349,776,800,000				
CMONITORINGSTATUS	0				
CCRITICALITY	1				
CARCHIVINGSTATUS	NONE				
CARCHIVINGSYSTEM	<null>				
CARCHIVINGMESSAGE	<null>				
CTIMESTAMP	1,349,767,824,346				
tbismt_documents	**Example**	**Internal Description**	**BIC Mapping**	**BIS Value**	**Frontend Description**
CCLIENTID	1				
CID	0000000050.Activity[0]		strTrackID$ & "." & strActivity$, strTrackID$	TRACK_ID.ACTIVITY	
CWFID	50		strTrackID$	TRACK_ID	
CDOCNUMBER	50		strTrackID$	TRACK_ID	
CDOCTYPE	SEEFX.Upload File		strToAdapter$	TO_ADAPTER	
CNUMBEROFLINES	<null>				
CFREETEXT	Uploading to SEE FX: file name is $[FROM_PATH]		strLogMessage$	LOG_MESSAGE	
CREFERENCES	<null>				
CFREE1	<null>		strTargetName$	TARGET_NAME	
CFREE2	<null>		strTargetPath$	TARGET_PATH	
CFREE3	<null>		strToName$	TO_NAME	
CFREE4	<null>		strToPath$	TO_PATH	
CFREE5	<null>		strToSubject$	TO_SUBJECT	

tbismt_workflow	Example	Internal Description	BIC Mapping	BIS Value	MT Frontend Description
CFREE6	<null>		strToID$	TO_ID	
CFREE7	<null>		strToUUID$	TO_UUID	
CFREE8	<null>		strDirection$ / strToAdapter$ (ERROR)	DIRECTION / TO_ADAPTER	
CFREE9	<null>		strToUUID$	TO_UUID	
CFREE10	<null>		strToID$	TO_ID	
CFREE11	<null>		strToSubject$	TO_SUBJECT	
CFREE12	<null>		strToPath$	TO_PATH	
CFREE13	<null>				
CFREE14	<null>				
CFREE15	<null>		strExecTime$	EXEC_TIME	
CFREE16	<null>				
CFREE17	<null>				
CFREE18	<null>				
CFREE19	<null>				
CFREE20	<null>				
CFREEDECIMAL1	<null>				
CFREEDECIMAL2	<null>				
CFREEDECIMAL3	<null>				
CFREEDECIMAL4	<null>				
CFREEDECIMAL5	<null>				
CERPSTATUS	<null>				
CLINK1	<null>				
CLINK2	<null>				
CLINK3	<null>				