NOKIA

BST Interface for Catalog
Release 19

Functional Description

Disclaimer

The information in this document applies solely to the hardware/software product ("Product") specified herein, and only as specified herein. Reference to "Nokia" later in this document shall mean the respective company within Nokia Group of Companies with whom you have entered into the Agreement (as defined below).

This document is intended for use by Nokia's customers ("You") only, and it may not be used except for the purposes defined in the agreement between You and Nokia ("Agreement") under which this document is distributed. No part of this document may be used, copied, reproduced, modified or transmitted in any form or means without the prior written permission of Nokia. If You have not entered into an Agreement applicable to the Product, or if that Agreement has expired or has been terminated, You may not use this document in any manner and You are obliged to return it to Nokia and destroy or delete any copies thereof.

The document has been prepared to be used by professional and properly trained personnel, and You assume full responsibility when using it. Nokia welcomes your comments as part of the process of continuous development and improvement of the documentation. This document and its contents are provided as a convenience to You. Any information or statements concerning the suitability, capacity, fitness for purpose or performance of the Product are given solely on an "as is" and "as available" basis in this document, and Nokia reserves the right to change any such information and statements without notice. Nokia has made all reasonable efforts to ensure that the content of this document is adequate and free of material errors and omissions, and Nokia will correct errors that You identify in this document. Nokia's total liability for any errors in the document is strictly limited to the correction of such error(s). Nokia does not warrant that the use of the software in the Product will be uninterrupted or error-free.

NO WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF AVAILABILITY, ACCURACY, RELIABILITY, TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, IS MADE IN RELATION TO THE CONTENT OF THIS DOCUMENT. IN NO EVENT WILL NOKIA BE LIABLE FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL OR ANY LOSSES, SUCH AS BUT NOT LIMITED TO LOSS OF PROFIT, REVENUE, BUSINESS INTERRUPTION, BUSINESS OPPORTUNITY OR DATA THAT MAY ARISE FROM THE USE OF THIS DOCUMENT OR THE INFORMATION IN IT, EVEN IN THE CASE OF ERRORS IN OR OMISSIONS FROM THIS DOCUMENT OR ITS CONTENT.

This document is Nokia proprietary and confidential information, which may not be distributed or disclosed to any third parties without the prior written consent of Nokia.

Nokia is a registered trademark of Nokia Corporation. Other product names mentioned in this document may be trademarks of their respective owners.

⚠ Important Notice on Product Safety

This product may present safety risks due to laser, electricity, heat, and other sources of danger.

Only trained and qualified personnel may install, operate, maintain or otherwise handle this product and only after having carefully read the safety information applicable to this product.

The safety information is provided in the Safety Information section in the "Legal, Safety and Environmental Information" part of this document or documentation set.

Nokia is continually striving to reduce the adverse environmental effects of its products and services. We would like to encourage you as our customers and users to join us in working towards a cleaner, safer environment. Please recycle product packaging and follow the recommendations for power use and proper disposal of our products and their components.

If you should have questions regarding our Environmental Policy or any of the environmental services we offer, please contact us at Nokia for any additional information.

1 About This Document

This document describes BST Interface for Catalog 19, which is compatible with Catalog release 19 and Business Service Tool release 19.

1.1 Audience

This document is intended for anyone who needs to use BST Interface for Catalog. The reader should be familiar with InstantLink, Business Service Tool and Catalog.

1.2 Terms and Concepts

The following sections list the abbreviations, terms and concepts used in the document.

1.2.1 Abbreviations

BST	Business Service Tool
NE	Network Element
UI	User Interface

1.2.2 Terminology

BST Interface for Catalog	An interface between Business Service Tool and Catalog. It allows BST to retrieve technical product definitions from Catalog.
Business Service Tool	A module that enables provisioning logic execution with InstantLink.
Catalog	A catalog that allows service providers to manage their service lifecycle and provides them with a centralised view of their product portfolio. It combines the commercial and technical view of offerings, and decomposes them into reusable elements.
InstantLink	A system for subscriber provisioning and service activation from OSS/BSS to the communications network. InstantLink receives requests from the OSS/BSS systems, translates the requests to network-element-specific commands and executes these commands. After execution of a request, InstantLink sends a response to the OSS/BSS.
item	Catalog allows user to create and maintain product, service or resource definitions as generic items. Items can be explicitly defined as products, services or any user defined term, or they can be left as generic items. Products and Services are defined by encapsulating and combining these items.
Operations and Business Support System, OSS/BSS	A set of programs that help an operator monitor, control, analyse and manage usage of a communications network. OSS/BSS include, for example, systems for customer care, order management, billing, relationship management, decision support, market analysis, fraud detection, traffic engineering and network planning. The individual properties of each OSS/BSS system determine what kind of requirements the customer sets for the Provisioning Systems, such as what services the Provisioning System should activate for subscribers.
product	A CRM or other BSS system level product that defines a sellable and marketable, business driven commercial offering.
service	A group of technical services that can be sold to a customer as part of a product.
state	A product's phase in its life-cycle. When a product traverses from development to testing and finally into a commercial product, its state changes accordingly from Working to Testing and Published. Catalog has a predefined state model which determines the state transitions.
technical library	An item in the Catalog model for which an Business Service Tool logic library should be called instead of sending a task (technical service).
technical service	A specification of a network level capability with attributes that define a network driven service.
technical service group	A group of technical services (and libraries) independent of each other that can be run in parallel.

1.3 Related Documentation

The following documents give you additional information on BST Interface for Catalog:

- BST Interface for Catalog Release Notes

- BST Interface for Catalog Installation Guide

For more information on InstantLink, see InstantLink documentation.

For more information on Business Service Tool, see Business Service Tool documentation.

For more information on Catalog, see Catalog documentation.

2 Interface Overview

This chapter provides general information on BST Interface for Catalog and the environment in which the interface is used.

BST Interface for Catalog is an add-on component to Business Service Tool that can be used to access Catalog from a provisioning logic. It includes BST steps that allow retrieving:

- product and service decomposition
- rollback information
- how to upgrade from one product to another from Catalog.

The steps are added to the step palette of Business Service Tool Logic Editor during the installation.

Additionally, the interface contains an in-memory cache that stores a snapshot of data recently retrieved from Catalog.

2.1 System Environment

Figure 1 illustrates the provisioning environment of Business Service Tool with Catalog.

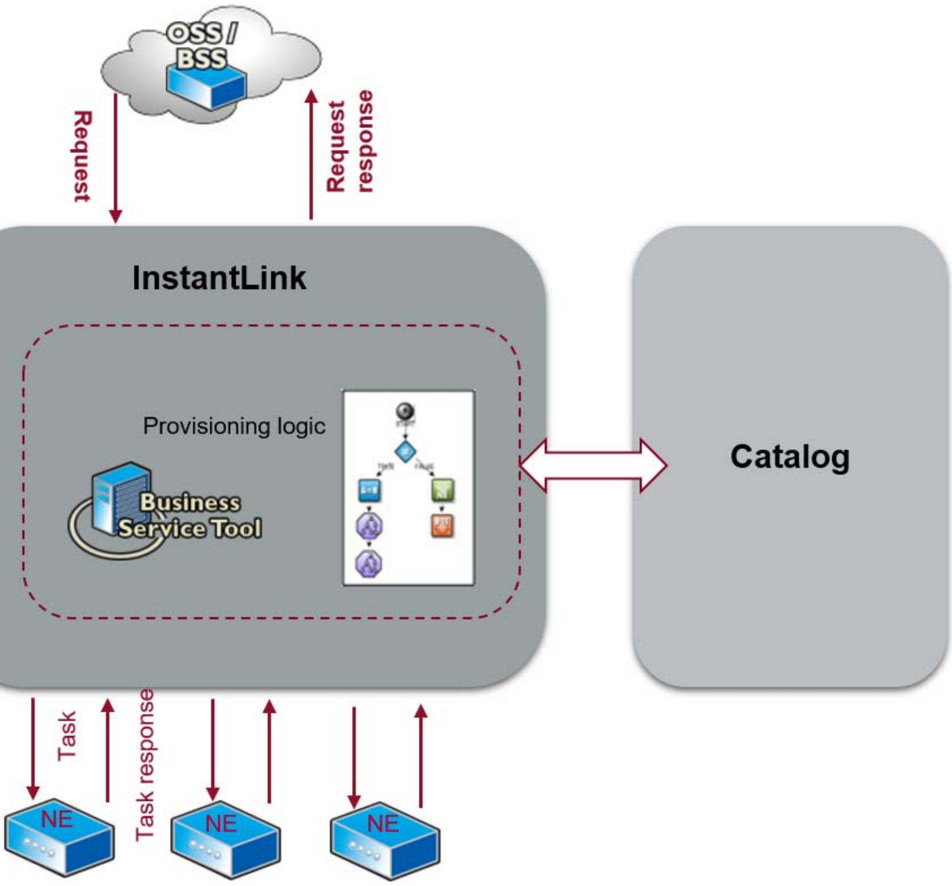


Figure 1. Business Service Tool provisioning environment with Catalog

The request process consists of the following phases:

1. The OSS/BSS system sends a request to InstantLink.
2. InstantLink stores the request and starts processing it.
3. InstantLink passes the request to Business Service Tool.
4. Business Service Tool receives the request, chooses the correct provisioning logic and starts processing the provisioning logic.
5. The provisioning logic requests a product or service specification from Catalog.
6. Catalog returns a list of attributes required for the requested product or service.
7. The provisioning logic fills in the attribute values.
8. The provisioning logic requests decomposition for the product or service from Catalog.
9. Catalog returns all attributes with values and dependencies required for the requested product or service.
10. The provisioning logic creates task parameters for each technical service and groups tasks according to dependency information.
11. Business Service Tool gives the group of tasks to InstantLink.
12. InstantLink sends the tasks to a network element.
13. The network element performs the tasks and sends the task response to InstantLink.
14. InstantLink sends the task response to Business Service Tool.
15. Business Service Tool continues processing the provisioning logic, one group of tasks after the other, repeating steps from 12 to 15 until all tasks have been processed.
16. Business Service Tool creates a request response.
17. Business Service Tool sends the request response to InstantLink.
18. InstantLink sends the request response to the OSS/BSS system.

2.2 Introduction to the Interface

The Catalog installation package includes the following:

- Java `jar` files for communication with Catalog
- `<installdir>/sas/bst/bst_cat/client.properties` and `<installdir>/sas/bst/bst_cat/jndi/jndi.properties` properties files for communication settings
- `<installdir>/sas/bst/extensions/bst_cat_extension.xml` file for configuring default transactions and item type on logic editor
- Business Service Tool steps with an Online Help are needed when BST interacts with Catalog
- Test request `<installdir>/sas/data/catalog_test_request` and Catalog model `catalog_test_items.xml` for verifying the installation

Figure 2 describes the test provisioning logic for verifying the interface as a graphical flowchart.

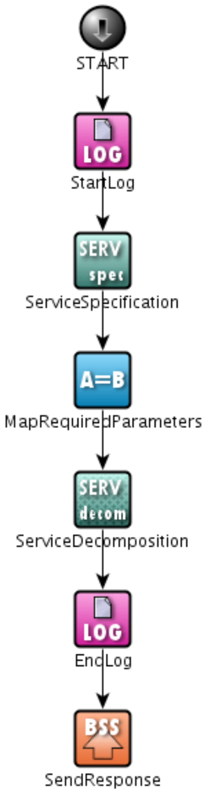


Figure 2. Test provisioning logic `catalog_test_logic`

The test logic consists of the following steps:

1. `Log` step writes a start message to the BST engine log.

2. `ServiceSpecification` step defines the product and operation which specification is fetched from Catalog.

3. `MapRequiredParameters` step maps the values for mandatory and optional parameters.

4. `ServiceDecomposition` step fetches the product decomposition from Catalog using the values defined in the `ServiceSpecification` step.

5. `EndLog` step writes an end message to the BST engine log.

6. `SendResponse` step creates a response message with product decomposition fetched from Catalog and stops logic execution.

Note The test logic shows how to fetch product decomposition from Catalog. However, it does not attempt to make any changes to the network. When the test request has been successfully processed by the test provisioning logic, the final status of the request is `Ready`. The request execution is terminated intentionally before making any changes to the network. The response contains product decomposition fetched from Catalog.

For more information on how to use examples, see *BST Interface for Catalog Installation Guide*.

3 Functionality

This chapter describes the functionality of BST Interface for Catalog.

3.1 Client Cache

BST Interface for Catalog client cache is an in-memory cache which stores a snapshot of product and service decompositions and execution orders of items. The cache provides fast access to decomposition and execution order data. If the data is found in the cache with the given criteria, the cache data is used instead of getting the data from Catalog.

The cache listens to state change notifications from Catalog, and if data in Catalog changes, the data in the cache must be updated accordingly. For example, if the state of a product or service changes from published to outdated or suspended or from testing or suspended to published, the decomposition entry and any execution order data containing the changed item is removed from the cache.

The cache configuration is located in the `<installdir>/sas/bst/bst_cat/ehcache.xml` file.The configuration contains an entry for each type of cached item. To tune the cache expiration, the following attributes can be configured:

- `timeToIdleSeconds` - The maximum number of seconds an item can exist in the cache without being accessed. The default value `604800` (7 days).
- `maxEntriesLocalHeap` - The maximum number of items that are kept in heap memory.

Information about items that are not in Catalog can also be cached. The functionality is configured with the `cacheNonExistingItems` property in `<installdir>/sas/bst/bst_cat/client.properties` file.

For any changes in the cache configuration to take effect, InstantLink must be restarted. The contents of the client cache are cleared during InstantLink System Refresh.

In some cases after a system refresh, there might be a significant queue of queries from BST Interface for Catalog towards Catalog. If there are several parallel queries issued for the same item, Catalog is put under unwanted and unnecessary pressure, and the related request processing slows down while each request thread is kept waiting in the queue. These kind of duplicate queries towards Catalog can be prevented by configuring the `cacheMaxWaitTime` property in the `<installdir>/sas/bst/bst_cat/client.properties` file.

Whenever an item is not found in the internal cache, BST Interface for Catalog checks if a query for the same item is already being made. If there is an ongoing query for the same item, the current thread waits a maximum of `cacheMaxWaitTime` milliseconds for the ongoing query to complete. Waiting is also stopped once the ongoing query has been completed. After the wait, the cache is checked again. If the item is still not found in the cache, a query towards Catalog is made.

3.2 Configuring Connection to Catalog

Connection between BST Interface for Catalog and Catalog is configured with `<installdir>/sas/bst/bst_cat/client.properties` and `<installdir>/sas/bst/bst_cat/jndi/jndi.properties` files. Changes in these files require a restart of InstantLink.

3.2.1 `jndi.properties` Properties File

The `jndi.properties` file is used by BST Interface for Catalog to set parameters that are required for communicating with Catalog.

As Catalog is installed separately on another server, Catalog uses TCP JMS connection to communicate with its client.

Table 1. Parameters in `jndi.properties` file

Parameter	Description
`java.naming.provider.url`	Catalog ActiveMQ URL in the following format: `tcp://<host>/<port>` **Default value:** `tcp://CATALOG_HOST:CATALOG_ACTIVEMQ_PORT` **Note** Values in this field are initially set when the installation script is executed.
`java.naming.factory.initial`	The Context Interface class that BST Interface for Catalog uses to communicate with the server. **Default value:** `org.apache.activemq.jndi.ActiveMQInitialContextFactory`
`topic.jms/topic/CatalogStateChange`	Registers topics in JNDI. **Default value:** `jms/topic/CatalogStateChange`
`connectionFactoryNames`	Specifies the JNDI name for the connection factory. **Default value:** `jms/cf/CatalogStateChange`

3.2.2 `client.properties` Properties File

The `client.properties` file is used by the BST Interface for Catalog for setting the Catalog authentication and configuration properties.

Table 2. Parameters in `client.properties` file

Parameter	Description
`catalog.protocol`	The Context Interface class that BST Interface for Catalog uses to communicate with the server. **Default value:** `HTTP`
`catalog.host`	The name of the host where Catalog is installed. **Default value:** `<hostname>` **Note** The value in this field is initially set when the installation script is executed.

Table 2. Parameters in `client.properties` file (Continued)

Parameter	Description
`catalog.port`	The port where Catalog is installed. **Default value:** `<port>` **Note** The value in this field is initially set when the installation script is executed.
`catalog.security`	The security property used for authenticating the communication session. For more information on the security property used, see *Catalog Api Auth Scheme* available in *Catalog Operational Client* documentation. Possible values: • `OFF` • `USERNAME` • `SESSION_ID` **Default value:** `SESSION_ID`
`catalog.username`	The Catalog client user name. **Default value:** `<username>` **Note** The value in this field is initially set when the installation script is executed.
`catalog.password`	The Catalog client user password. The password is SHA-1 encoded. **Default value:** `<SHA-1 encoded password>` **Note** The value in this field is initially encoded and set when the installation script is executed. If the password has changed, see Section 3.4 *Configuring BST Interface for Catalog User Name and Password* for more details.
`union_technical_services`	Determines whether a union of technical services is created during decomposition. If set to `true`, duplicate technical services within a technical service group are removed. A technical service is considered a duplicate if the same technical service with the same attributes is already found in the group. **Default value:** `true`
`all_lifecycle_states`	Specifies the comma-separated list of Catalog lifecycle states to be included when the value is `ALL`. For more information on the value used by Catalog, see *All Lifecycle States*, available in *Catalog Operational Client* documentation. **Default value:** `Published,Testing,Suspended`

Table 2. Parameters in `client.properties` file (Continued)

Parameter	Description
`pool_maxActive`	Determines the maximum number of active connection pool objects that can be allocated at a given time. **Default value**: `20` **Note** The value in this field applies to each running instance of InstantLink Service Module Engine.
`pool_maxIdle`	Determines the maximum number of idle connection pool objects that can be allocated at a given time. **Default value:** `20` **Note** The value in this field applies to each running instance of InstantLink Service Module Engine.
`pool_whenExhausted`	Determines the action that is taken when pool size is exhausted. **Possible values:** • `0` (login fails if the pool size is exhausted) • `1` (login is blocked until a new connection is available) • `2` (the pool size grows when it is exhausted) **Default value:** `1` **Note** The value in this field applies to each running instance of InstantLink Service Module Engine.
`pool_maxWait`	Maximum amount of time in milliseconds that the login connection waits until a new connection is available, and when the pool is exhausted with the value of `1` (`pool_whenExhausted=1`). A value smaller than `0` indicates that the login is blocked immediately until a new connection is available. **Default value:** `-1` **Note** The value in this field applies to each running instance of InstantLink Service Module Engine.
`pool_maxConnectionsPerHost`	Maximum number of connections per host. **Default value:** `20` **Note** The value in this field applies to each running instance of InstantLink Service Module Engine.
`connectionTimeout`	The HTTP connection timeout when reading data from Catalog, in seconds. **Default value**: `60`
`cacheNonExistingItems`	Specifies whether information about items that are not in Catalog is cached or not. Possible values: • `true` • `false` Default value: `false`

Table 2. Parameters in `client.properties` file (Continued)

Parameter	Description
`cacheMaxWaitTime`	Maximum time in milliseconds to wait for an ongoing query to finish for the same Catalog item before querying it again. Default value: `30000`
`connectionRetryCount`	Specifies how many times to retry a broken Catalog connection before failing the operation. Possible values: `0` = no retry `-1` = try indefinitely `<n>` = number of retry attempts Default value: `1440`
`connectionRetryInterval`	Interval in seconds how often a reconnect attempt is launched when Catalog connection is broken. Default value: `60`
`useRemoteItemGroups`	Specifies if remote item groups are fetched from Catalog along with decomposition queries. Possible values: • `true` • `false` Default value: `false`

3.2.3 Configuring Retry for Broken Catalog Connections

BST Interface for Catalog has functionality for retrying failed operations when the connection to Catalog is lost. When the connection to Catalog is noticed to be lost, the connection is marked as broken, an alarm is sent and a reconnect timer is started. All new incoming operations wait for the connection to be back up again. Operations remain waiting until their individual wait time has elapsed or the connection to Catalog has been re-established.

The reconnect timer lets one waiting operation to retry the connection to Catalog at an interval defined in property `connectionRetryInterval`. The default time to wait for the connection is 24 hours. If the connection is successfully re-established, all other waiting operations are released from waiting and they may proceed. If the connection is still broken, the retry operation is returned to waiting state. Maximum wait time in seconds for each operation can be estimated by multiplying `connectionRetryInterval` with `connectionRetryCount`.

There are two special values for property `connectionRetryCount`:

- Value `0` means that connection retry functionality is not used at all
- Value `-1` means that operations waits forever until the connection is re-established

Any waiting operation blocks the corresponding Service Module Engine thread and thus the amount of waiting operations can be up to Service Module Engine thread count. When all available Service Module Engine threads are waiting for Catalog connection, no other requests in the Service Module Engine may be executed.

3.3 Configuring Default Transactions and Item Types

Catalog allows configuring default transactions and items types freely. The same values should be configured to BST Interface for Catalog to avoid accidental configuration errors.

In Catalog Default Transactions are defined in Operational Client under the **System parameter** tab. The supported transactions are listed in parameter `DEFAULT TRANSACTIONS`. The item types are defined in Operation Client under the **Item Types** tab.

In BST Logic Editor both `ServiceSpecification` and `ErrorDecomposition` steps have drop-down lists for item type and operation. Default items can be configured for the lists in `<installdir>/sas/bst/extensions/bst_cat_ext.xml` file. Changes in the file require a restart of InstantLink.

The XML file has a section named additional-parameters that contains two parameters described in *Table 3*.

Table 3. `bst_cat_ext.xml` file properties

Parameter name	Default value	Description
`ui_default_item_types`	`Product,Service,Technical Service,Technical Library`	A comma-separated list of values visible in the item type list.
`ui_default_transactions`	`Create,Modify,Delete, Display`	A comma-separated list of values visible in the operation list.

3.4 Configuring BST Interface for Catalog User Name and Password

BST Interface for Catalog user name and password must be the same as those that have been configured for Catalog. For more information on Catalog username and password, see *Catalog Operational Users Guide, Chapter, Administration screen.*

To configure username and password

1. Source the `.profile` file with the following command:

```
. <installdir>/sas/.profile
```

2. Shut down InstantLink with the following command:

```
ctl_control stop
```

3. Encrypt a password, using SHA-1 with the following command:

```
encrypt_password.ksh <PASSWORD>
```

Example:

```
encrypt_password.ksh pass1234
789b49606c321c8cf228d17942608eff0ccc4171
```

4. Enable write access for the `client.properties` file with the following command:

   ```
   chmod +w $SASPATH/sas/bst/bst_cat/client.properties
   ```

5. Update the user name and password to the `client.properties` file and save the changes.

   ```
   catalog.username=<username>
   ```

   ```
   # This property specifies the SHA-1 encoded password for
   the Catalog API
   catalog.password=<encrypted password>
   ```

   Example:

   ```
   catalog.username=admin
   catalog.password=789b49606c321c8cf228d17942608eff0ccc417
   1
   ```

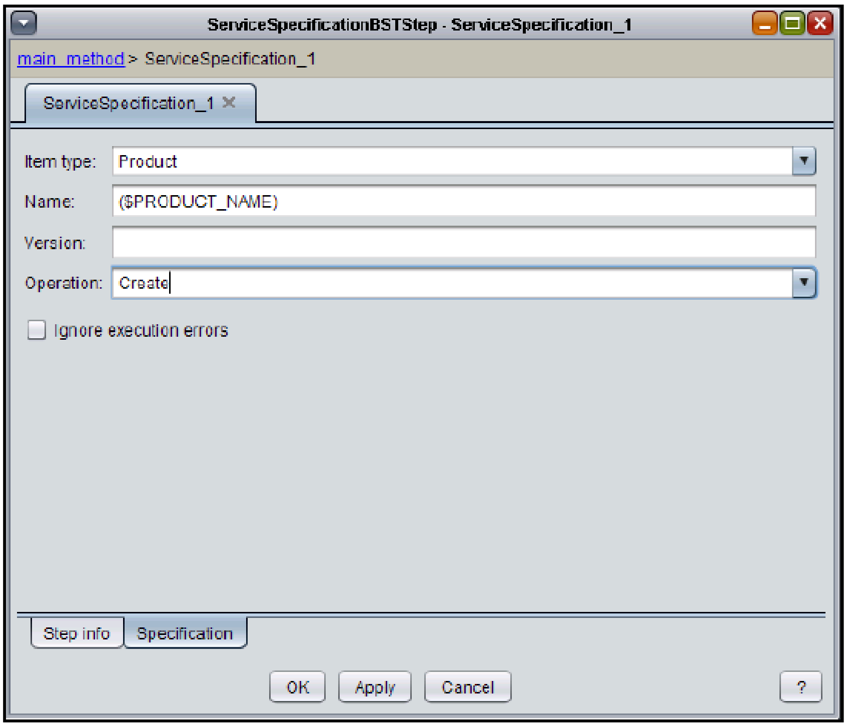
6. Restart InstantLink with the following command:

   ```
   ctl_control start
   ```

3.5 ServiceSpecification Step

Use the `ServiceSpecification` step SERV spec to retrieve the product or service specification from Catalog. The `ServiceSpecification` step verifies, whether the BST parameter pool has all the necessary parameters to process the decomposition for a given item. If not, it returns a list of missing mandatory parameters.

The `ServiceSpecification` step must be run before the `ServiceDecomposition` step can be used to retrieve the actual `decomposition`. *Figure 3* illustrates the `ServiceSpecification` step.



ServiceSpecificationBSTStep - ServiceSpecification_1
main_method > ServiceSpecification_1
ServiceSpecification_1 ✕
Step info
Specification
OK
Apply
Cancel
?

Item type: Product

Name: ($PRODUCT_NAME)

Version:

Operation: Create

□ Ignore execution errors

Figure 3. `ServiceSpecification` step

Table 4 lists the requested input parameters. In the M/O column, M stands for a mandatory option and O for an optional one.

Table 4. `ServiceSpecification` input parameters

Parameter	UI field	Description	M/O
`SC_STEP_ENTITY_TYPE`	Item type	Item type in Catalog. **Example values:** • `Product` • `Service` • `Technical Service` • `Technical Library`	`M`
`SC_STEP_ENTITY_NAME`	Name	Name of a product or service in Catalog, for example, the name of a product.	`M`
`SC_STEP_ENTITY_VERSION`	Version	Version of the item. If the value is empty, the latest published version is used. **Value format:** `<major version>.<minor version>` **Example:** `1.0`	`O`

Table 4. `ServiceSpecification` input parameters (Continued)

Parameter	UI field	Description	M/O
`SC_STEP_OPERATION`	Operation	Requested operation in Catalog. **Example values:** • `Create` • `Modify` • `Delete` • `Display`	`M`
`SC_STEP_ENTITY_ID`		Integer number that uniquely identifies an item and its version. In Catalog UI the parameter is called External ID. The parameter does not have its own field in the **Specification** tab of the step but you can define it, for example, with the `MapParameters` step. If `SC_STEP_ENTITY_ID` is defined, also Search type (`SC_STEP_OPERATION`) must be defined. If both Name (`SC_STEP_ENTITY_NAME`) and `SC_STEP_ENTITY_ID` are defined, Name is used. **For example:** Product “Chat a lot” version 1.0 is identified with `SC_STEP_ENTITY_ID 782` Product “Chat a lot” version 1.1 is identified with `SC_STEP_ENTITY_ID 1025` If `SC_STEP_ENTITY_ID` is defined, Search type (`SC_STEP_OPERATION`) must also be defined. However, Type (`SC_ENTITY_TYPE`), Name (`SC_STEP_ENTITY_NAME`) and Version (`SC_STEP_ENTITY_VERSION`) must be left empty.	`O`

Table 4. `ServiceSpecification` input parameters (Continued)

Parameter	UI field	Description	M/O
`SC_STEP_SERVICE_STATE`		State of a product or service. The parameter does not have its own field in the UI but you can define it, for example, with the `MapParameters` step. **Example values:** • `ALL` • `Testing` • `Suspended` • `Published` • `ANY` Value `ALL` returns the latest version that is in one of the states defined in Catalog `all_lifecycle_states` parameter in the `client.proerties` file. For example, if the value of `all_ lifecycle_states` is "Testing, Published, Suspended", the latest version that is in one of these states is returned.	O
`SC_IGNORE_ERRORS`	Ignore execution errors	If enabled, `SC_ERROR_CODE` parameter is put to the parameter pool in case of an error. If disabled, BST logic execution stops. **Possible values:** • `TRUE` • `FALSE` (Default value)	O
`SC_STEP_DEPENDENCIES_EXCLUSIONS`		Defines whether this step adds parameters for capabilities, dependencies and exclusions into the parameter pool. **Possible values:** • `TRUE` (Default value) • `FALSE` The parameter does not have its own field in the UI, but you can define it, for example, with the `MapParameters` step.	O

Table 5 lists the output parameters that the `ServiceSpecification` step returns to the parameter pool.

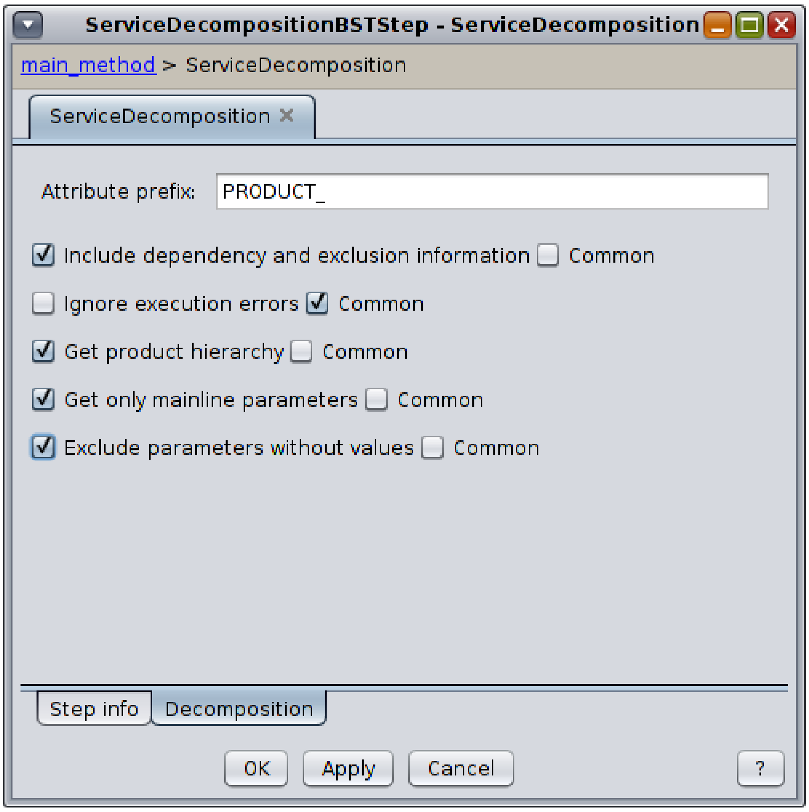
Table 5. `ServiceSpecification` step output parameters

Parameter	Description
`SC_MISSING_MANDATORY`	List of mandatory product/service attributes. Without these parameters it is not possible to perform the operation defined by parameter `SC_STEP_ENTITY_OPERATION.` **Example value**: `SC_MISSING_MANDATORY=MSISDN,IMSI`
`SC_MISSING_OPTIONAL`	List of optional product/service attributes. **Example value:** `SC_MISSING_OPTIONAL=CLIR,CLIP`
`SC_CAPABILITIES`	A comma-separated list of capabilities defined for the requested product/service. This parameter is not generated if there are no capabilities defined for the product/service. **Example value:** `SC_CAPABILITIES=ADSL,BASIC_GSM,VMS,SMS`
`SC_DEPENDENCIES`	A comma-separated list of dependencies defined for the requested product/service. This parameter is not generated if there are no dependencies defined for the product/service. **Example value:** `SC_DEPENDENCIES=INTERNET_ACCESS,MOBILE_VOICE`
`SC_EXCLUSIONS`	A comma-separated list of exclusions defined for the requested product/service. This parameter is not generated if there are no exclusions defined for the product/service. **Example value:** `SC_EXCLUSIONS=VSDL`

3.6 ServiceDecomposition Step

Use the `ServiceDecomposition` step to retrieve product or service decomposition from Catalog. The product or service must be verified with the `ServiceSpecification` step first. Only then it is possible to retrieve the decomposition with the `ServiceDecomposition` step. By default this step creates parameters only for the technical services/libraries of the product/service. Optionally, parameters for full product hierarchy can be created.

Figure 4 illustrates the `ServiceDecomposition` step.



ServiceDecompositionBSTStep - ServiceDecomposition

main_method > ServiceDecomposition

Attribute prefix: PRODUCT_

☑ Include dependency and exclusion information □ Common

□ Ignore execution errors ☑ Common

☑ Get product hierarchy □ Common

☑ Get only mainline parameters □ Common

☑ Exclude parameters without values □ Common

Figure 4. `ServiceDecomposition` step

Table 6 lists the requested input parameters for the step.

Table 6. `ServiceDecomposition` step input parameters

Parameter	UI field	Description	M/O
`SC_STEP_PARAMETER_PREFIX`	Attribute prefix	Prefix to be used in product/service/ technical service /technical library parameters. The prefix allows identifying which parameters in the parameter pool result from the decomposition done by Catalog.	O
`SC_STEP_DEPENDENCIES_EXCLUSIONS`	Include dependency and exclusion information	Defines whether this step creates parameters for capabilities, dependencies and exclusions into the parameter pool. There is also a parameter for missing dependencies on decomposition level. Not applicable for product hierarchy decomposition. **Default value:** `false`	O

Table 6. `ServiceDecomposition` step input parameters (Continued)

Parameter	UI field	Description	M/O
`SC_IGNORE_ERRORS`	Ignore execution errors	If enabled, `SC_ERROR_CODE` parameter is put to the parameter pool in case of an error. If disabled, Business Service Tool logic execution stops when an error occurs. **Possible values:** • `TRUE` • `FALSE` (Default value)	O
`SC_FULL_DECOMPOSITION`	Get product hierarchy	If enabled, the step creates parameters for full product hierarchy. **Possible values:** • `TRUE` • `FALSE` (Default value)	O
`SC_MAINLINE_PARAMS_ONLY`	Get only mainline parameters	If enabled, the step only returns parameters for creating the mainline in the Order Management Gantt chart. **Note** This parameter can only be used when **Get product hierarchy** has been enabled. **Possible values:** • `TRUE` • `FALSE` (Default value)	O
`SC_EXCLUDE_PARAMS_WITH_NO_VALUE`	Exclude parameters without values	If enabled, decomposition parameters that do not have values are not added to the parameter pool. Possible values: • `TRUE` • `FALSE` (Default value)	O

3.6.1 Technical Service/Library Level Decomposition

When you fetch the lowest level of the product hierarchy, the decomposition consists of a list of technical services/libraries and information on the order in which they must be processed in the network.

Table 7 lists the output parameters for technical service/library level decomposition.

Table 7. `ServiceDecomposition` step output parameters, technical service/ library level decomposition

Parameter	Description
`<prefix><y>_<z>_<parameter name>=<parameter value>`	List of product/service parameters and values from Catalog and capability library. y=technical service group (task group) z=technical service/library (task/library) The technical service group number defines the order in which the tasks must be sent to InstantLink. If there are several tasks within a group, they can be sent using the `ParallelSend2` BST step. **Example value:** `PRODUCT1_1_NE_TYPE=HLR` **Note** The technical service group may contain a mixture of technical services and libraries. **Note** The parameter value may contain a parameter reference to a parameter value from a previous technical service/library. Example: `PROD_2_1_OLD_MSISDN=($MSISDN)`
`<prefix>CAPABILITIES`	List of capabilities on decomposition level. **Example value:** `PRODUCT_CAPABILITIES=ADSL, BASIC_GSM, VMS, SMS, BLACKBERRY`
`<prefix>DEPENDENCIES`	List of dependencies on decomposition level. **Example value:** `PRODUCT_DEPENDENCIES=INTERNET_ACCESS, MOBILE_VOICE`
`<prefix>EXCLUSIONS`	List of exclusions on decomposition level. **Example value:** `PRODUCT_EXCLUSIONS=VSDL`
`<prefix>MISSING_DEPENDENCIES`	List of missing dependencies on decomposition level. **Example value:** `PRODUCT_MISSING_DEPENDENCIES=MOBILE_VOICE`

`<prefix><y>_<z>_CAPABILITIES`	List of capabilities on lowest (technical service/library) level.
	y=technical service group (task group)
	z=technical service/library (task/library)
	Example value:
	`PRODUCT_1_1_CAPABILITIES=ADSL`

Table 7. `ServiceDecomposition` step output parameters, technical service/library level decomposition (Continued)

Parameter	Description

`<prefix><y>_<z>_DEPENDENCIES`	List of dependencies on lowest (technical service/library) level.
	y=technical service group (task group)
	z=technical service/library (task/library)
	Example value:
	`PRODUCT_1_1_DEPENDENCIES=INTERNET_ACCESS`

`<prefix><y>_<z>_EXCLUSIONS`	List of exclusions on lowest (technical service/library) level.
	y=technical service group (task group)
	z=technical service/library (task/library)
	Example value:
	`PRODUCT_1_1_EXCLUSIONS=VDSL`

`<prefix><y>_<z>_TS_NAME`	Name of the lowest level item (technical service/library) in Catalog.
	y=technical service group (task group)
	z=technical service/library (task/library)
	Example value:
	`PRODUCT_1_1_TS_NAME=HLR_voice`
	If several lowest level items are grouped in Catalog, the group is passed as one item to BST and `TS_NAME` contains the name of the group.

`<prefix><y>_<z>_TS_DURATION`	Duration of the lowest level item (technical service/library) in Catalog, defined in milliseconds.
	y=technical service group (task group)
	z=technical service/library (task/library)
	If there is no duration defined for the item in Catalog, the default value `-1` is returned.
	Example value:
	`PRODUCT_1_1_TS_DURATION=3600000`

`<prefix><y>_<z>_TS_VERSION`	Version of the lowest level item (technical service/library) in Catalog.
	y=technical service group (task group)
	z=technical service/library (task/library)
	Example value:
	`PRODUCT_1_1_TS_VERSION=4.1`

`<prefix><y>_<z>_TS_TYPE`	Type of the lowest level item (technical service/technical library) in Catalog. y=technical service group (task group) z=technical service/library (task/library) **Possible values:** `GTS` - remote item group `TS` - technical service `TL` - technical library

Table 7. `ServiceDecomposition` step output parameters, technical service/ library level decomposition (Continued)

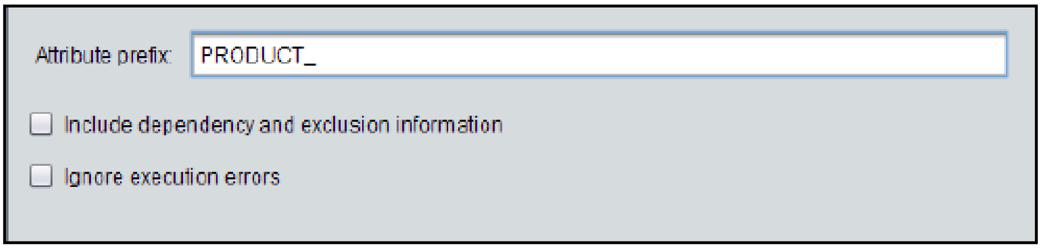
Parameter	Description

`<prefix><y>_<z>_ERROR_DATA`	The identifier to be given to the error decomposition step in case of a failure in technical service/library. Used for fetching the error (rollback) decomposition.
	y=technical service group (task group)
	z=technical service/library (task/library)

`SC_GROUP_COUNT`	The number of technical service groups in the parameter pool. **Example value:** `SC_GROUP_COUNT=2`

`SC_GROUP_SIZE_<y>`	The number of technical services/libraries within each group.
	y=Technical service group identifier
	Example value:
	`SC_GROUP_SIZE_1=3`
	`SC_GROUP_SIZE_2=5`

Example:


Attribute prefix: PRODUCT_

□ Include dependency and exclusion information

□ Ignore execution errors

This example provides the following parameters to the parameter pool:

```
PRODUCT_1_1_BASIC1=B1F
PRODUCT_1_1_ERROR_DATA=8f5a5a55ae91c1c03e62520b54869a0bc58ac55
e
PRODUCT_1_1_IMSI1=010123456789
PRODUCT_1_1_MSISDN1=BS11
PRODUCT_1_1_NE_TYPE=HLR
PRODUCT_1_1_REQ_OBJ=1
PRODUCT_1_1_REQ_TYPE=1
PRODUCT_1_1_TASK_TYPE=create
PRODUCT_1_1_TS_NAME=TS_SMS
PRODUCT_1_1_TS_DURATION=-1
PRODUCT_1_1_TS_TYPE=TS
PRODUCT_1_1_TS_VERSION=1.1
PRODUCT_1_2_ERROR_DATA=7d16367f1d7637592e281261f20c8e66369bd98
c
```

```
PRODUCT_1_2_IMSI1=010123456789
PRODUCT_1_2_MSISDN1=BS11
PRODUCT_1_2_NE_TYPE=VMS
PRODUCT_1_2_REQ_OBJ=1
PRODUCT_1_2_REQ_TYPE=1
PRODUCT_1_2_TASK_TYPE=create
PRODUCT_1_2_TS_NAME=TS_VOICEMAIL
PRODUCT_1_2_TS_DURATION=-1
PRODUCT_1_2_TS_TYPE=TS
PRODUCT_1_2_TS_VERSION=1.1
PRODUCT_1_2_VMS_NUMBER=358507770001
PRODUCT_1_3_BASIC1=B1F
PRODUCT_1_3_ERROR_DATA=efd8bec3ec62e75e5a2f879f65d09476cd6214f
c
PRODUCT_1_3_IMSI1=010123456789
PRODUCT_1_3_MSISDN1=BS11
PRODUCT_1_3_NE_TYPE=HLR
PRODUCT_1_3_REQ_OBJ=1
PRODUCT_1_3_REQ_TYPE=1
PRODUCT_1_3_TASK_TYPE=create
PRODUCT_1_3_TS_NAME=TS_VOICE
PRODUCT_1_3_TS_DURATION=-1
PRODUCT_1_3_TS_TYPE=TS
PRODUCT_1_3_TS_VERSION=1.1
```

3.6.2 Product Hierarchy Decomposition

When the `ServiceDecomposition` step is used to get full product hierarchy, the response contains the full decomposition of the products/services/technical services/ libraries belonging to the matched product/service. The structure is represented as a "tree" using indexed groups and indexes inside groups. Note that there may be multiple levels in the tree, depending on the Catalog model of the fetched product/ service.

The parameters contain the group index/indexes and the position inside the group. Products/services/technical services/libraries/selection groups in a group can be executed in parallel. Each group has a parameter indicating the number of products/ services/technical services/libraries/selection groups inside the group: `GROUP_SIZE`. For example, `PRODUCT_2_GROUP_SIZE=3` indicates that the second group under the requested product/service contains three products, services, technical services/ libraries or selection groups.

Each sublevel of products, services, technical services/libraries or selection groups appends a new "`<group index>_<position inside the group>_`" part into the parent level prefix. For example, the first service in the first group in the fetched product/service would use "`PRODUCT_1_1_`" as prefix. The first technical service belonging to the first group of the mentioned service would use "`PRODUCT_1_1_1_1_`" as a prefix for all its parameters ("`PRODUCT_`" used here as the prefix for the full product decomposition).

Table 8. Output parameters, full product hierarchy decomposition

Parameter	Description
`<prefix><y>_<z>_NAME`	The name of the product/service in Catalog. y=group index/indexes z=position inside the group **Example value:** `Voice Only`
`<prefix><y>_<z>_ACTIVITY_NAME`	Activity name for the operation as defined in Catalog. May have an empty value. y=group index/indexes z=position inside the group **Example value:** `Create HLR subscriber`
`<prefix><y>_<z>_TYPE`	User defined type of the product/service/technical service/library. y=group index/indexes z=position inside the group **Example value:** `Product`
`<prefix><y>_<z>_ITEM_TYPE`	The type of the product/service/technical service/ library. y=group index/indexes z=position inside the group Possible values: • `LOCAL` (product or service) • `REMOTE` (technical service/library) • `SELECTION_GROUP` (selection group)
`<prefix><y>_<z>_VERSION`	The version of the product/service/technical service/ library. y=group index/indexes z=position inside the group **Example value:** **`1.0`**
`<prefix><y>_<z>_TRANSACTION_DURATION`	Duration of the operation for product/service/technical service/library, defined in millliseconds. y=group index/indexes z=position inside the group If there is no duration defined for the item in Catalog, the default value `-1` is returned. **Example value:** `3600000`

Table 8. Output parameters, full product hierarchy decomposition (Continued)

Parameter	Description
`<prefix><y>_<z>_TRANSACTION_NAME`	The operation name of the product/service/technical service/library. y=group index/indexes z=position inside the group **Example value:** `Create`
`<prefix><y>_<z>_UNDO_TRANSACTION_NAME`	The (rollback) operation name for the product/service/technical service/library. y=group index/indexes z=position inside the group **Example value:** `Delete`
`<prefix><y>_<z>_CAPABILITIES`	A comma-separated list of the capabilities of the product/service/technical service/library This parameter may also have an empty value. y=group index/indexes z=position inside the group
`<prefix><y>_<z>_DEPENDENCIES`	A comma-separated list of dependencies of the product/service/technical service/library. This parameter may also have an empty value. y=group index/indexes z=position inside the group
`<prefix><y>_<z>_EXCLUSIONS`	A comma-separated list of exclusions defined for the product/service/technical service/library. This parameter may also have an empty value. y=group index/indexes z=position inside the group
`<prefix><y>_<z>_GROUP_COUNT`	The number of groups under this product/service. Technical services/libraries do not have this parameter. y=group index/indexes z=position inside the group
`<prefix><y>_<z>_ERROR_DATA`	The identifier to be given to the error decomposition step in case of a failure in technical service/library. Used for fetching the error (rollback) decomposition. y=group index/indexes z=position inside the group **Note** This parameter is present only in case of a technical service or technical library (`ITEM_TYPE=REMOTE`).

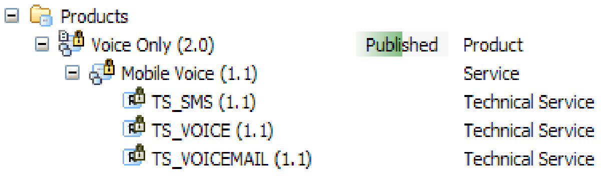
The other parameters from the Catalog model have additional prefixes. Prefixes are added to the parameter name as follows:

- `"I_"` for item parameters

- "T_" for transaction parameters
- "M_" for message parameters
  - "IN_" for input message parameters
  - "OUT_" for output message parameters. This is shown only for parameters that have a default value.
  - "OUT_PARAMS_" contains a comma-separated list of the item's output parameters.
  - "MAP_OUT_<PARAM_NAME>" contains mapping information for the item's output message parameter. The value of this parameter contains a target name for the parameter. This will be present when the output message parameter of an item must be mapped to another name on the parent item's output.

Example:

Catalog has a product 'Voice Only' with a service 'Mobile Voice'. The service contains three technical services (TS_SMS, TS_VOICE and TS_VOICEMAIL) that can be run in parallel.



⊟ Products		
⊟ Voice Only (2.0)	Published	Product
⊟ Mobile Voice (1.1)		Service
TS_SMS (1.1)		Technical Service
TS_VOICE (1.1)		Technical Service
TS_VOICEMAIL (1.1)		Technical Service

Figure 5. Product ‘Voice Only’ in Catalog

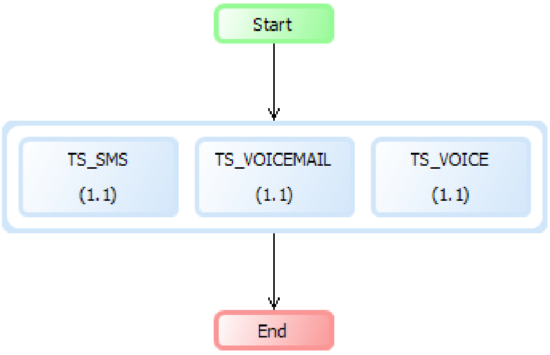


Figure 6. Technical services within service ‘Mobile Voice’

Full product hierarchy decomposition of product 'Voice Only' generates the following parameters to the pool:

```
PRODUCT_1_1_1_1_CAPABILITIES=TS_SMS
PRODUCT_1_1_1_1_DEPENDENCIES=TS_VOICE
PRODUCT_1_1_1_1_EXCLUSIONS=
PRODUCT_1_1_1_1_ITEM_TYPE=REMOTE
PRODUCT_1_1_1_1_I_NE_TYPE=HLR
PRODUCT_1_1_1_1_I_REQ_OBJ=1
PRODUCT_1_1_1_1_M_IN_BASIC1=B1F
PRODUCT_1_1_1_1_M_IN_IMSI1=010123456789
PRODUCT_1_1_1_1_M_IN_MSISDN1=BS11
PRODUCT_1_1_1_1_M_OUT_PARAMS=SERVICE_ID
PRODUCT_1_1_1_1_M_MAP_OUT_SERVICE_ID=SMS_SERVICE_ID
PRODUCT_1_1_1_1_NAME=TS_SMS
PRODUCT_1_1_1_1_ACTIVITY_NAME=Create SMS
PRODUCT_1_1_1_1_TRANSACTION_DURATION=-1
PRODUCT_1_1_1_1_TRANSACTION_NAME=CREATE
PRODUCT_1_1_1_1_TYPE=Technical Service
PRODUCT_1_1_1_1_T_REQ_TYPE=1
PRODUCT_1_1_1_1_T_TASK_TYPE=create
PRODUCT_1_1_1_1_UNDO_TRANSACTION_NAME=DELETE
PRODUCT_1_1_1_1_VERSION=1.1
PRODUCT_1_1_1_1_ERROR_DATA=8f5a5a55ae91c1c03e62520b54869a0bc58
ac55e
PRODUCT_1_1_1_2_CAPABILITIES=
PRODUCT_1_1_1_2_DEPENDENCIES=
PRODUCT_1_1_1_2_EXCLUSIONS=
PRODUCT_1_1_1_2_ITEM_TYPE=REMOTE
PRODUCT_1_1_1_2_I_NE_TYPE=VMS
PRODUCT_1_1_1_2_I_REQ_OBJ=1
PRODUCT_1_1_1_2_M_IN_IMSI1=010123456789
PRODUCT_1_1_1_2_M_IN_MSISDN1=BS11
PRODUCT_1_1_1_2_M_IN_VMS_NUMBER=358507770001
PRODUCT_1_1_1_2_M_OUT_PARAMS=
PRODUCT_1_1_1_2_NAME=TS_VOICEMAIL
PRODUCT_1_1_1_2_ACTIVITY_NAME=Create voicemail
PRODUCT_1_1_1_2_TRANSACTION_DURATION=-1
PRODUCT_1_1_1_2_TRANSACTION_NAME=CREATE
PRODUCT_1_1_1_2_TYPE=Technical Service
PRODUCT_1_1_1_2_T_REQ_TYPE=1
PRODUCT_1_1_1_2_T_TASK_TYPE=create
PRODUCT_1_1_1_2_UNDO_TRANSACTION_NAME=DELETE
PRODUCT_1_1_1_2_VERSION=1.1
PRODUCT_1_1_1_2_ERROR_DATA=7d16367f1d7637592e281261f20c8e66369
bd98c
PRODUCT_1_1_1_3_CAPABILITIES=TS_VOICE
PRODUCT_1_1_1_3_DEPENDENCIES=
PRODUCT_1_1_1_3_EXCLUSIONS=
PRODUCT_1_1_1_3_ITEM_TYPE=REMOTE
PRODUCT_1_1_1_3_I_NE_TYPE=HLR
PRODUCT_1_1_1_3_I_REQ_OBJ=1
```

```
PRODUCT_1_1_1_3_M_IN_BASIC1=B1F
PRODUCT_1_1_1_3_M_IN_IMSI1=010123456789
PRODUCT_1_1_1_3_M_IN_MSISDN1=BS11
PRODUCT_1_1_1_3_M_OUT_PARAMS=SERVICE_ID
PRODUCT_1_1_1_3_M_MAP_OUT_SERVICE_ID=VOICE_SERVICE_ID
PRODUCT_1_1_1_3_NAME=TS_VOICE
PRODUCT_1_1_1_3_ACTIVITY_NAME=Create mobile voice
PRODUCT_1_1_1_3_TRANSACTION_DURATION=-1
PRODUCT_1_1_1_3_TRANSACTION_NAME=CREATE
PRODUCT_1_1_1_3_TYPE=Technical Service
PRODUCT_1_1_1_3_T_REQ_TYPE=1
PRODUCT_1_1_1_3_T_TASK_TYPE=create
PRODUCT_1_1_1_3_UNDO_TRANSACTION_NAME=DELETE
PRODUCT_1_1_1_3_VERSION=1.1
PRODUCT_1_1_1_3_ERROR_DATA=efd8bec3ec62e75e5a2f879f65d09476cd6
214fc
PRODUCT_1_1_1_GROUP_SIZE=3
PRODUCT_1_1_CAPABILITIES=Mobile
Voice,TS_SMS,TS_VOICE,TS_VOICEMAIL
PRODUCT_1_1_DEPENDENCIES=
PRODUCT_1_1_EXCLUSIONS=
PRODUCT_1_1_GROUP_COUNT=1
PRODUCT_1_1_ITEM_TYPE=LOCAL
PRODUCT_1_1_M_IN_BASIC1=B1F
PRODUCT_1_1_M_IN_IMSI1=010123456789
PRODUCT_1_1_M_IN_MSISDN1=BS11
PRODUCT_1_1_M_IN_VMS_NUMBER=358507770001
PRODUCT_1_1_M_OUT_PARAMS=SMS_SERVICE_ID,VOICE_SERVICE_ID
PRODUCT_1_1_NAME=Mobile Voice
PRODUCT_1_1_ACTIVITY_NAME=Create mobile voice service
PRODUCT_1_1_TRANSACTION_DURATION=60000
PRODUCT_1_1_TRANSACTION_NAME=CREATE
PRODUCT_1_1_TYPE=Service
PRODUCT_1_1_UNDO_TRANSACTION_NAME=DELETE
PRODUCT_1_1_VERSION=1.1
PRODUCT_1_GROUP_SIZE=1
PRODUCT_CAPABILITIES=TS_VOICE,Voice Only
PRODUCT_DEPENDENCIES=
PRODUCT_EXCLUSIONS=
PRODUCT_GROUP_COUNT=1
PRODUCT_ITEM_TYPE=LOCAL
PRODUCT_M_IN_BASIC1=B1F
PRODUCT_M_IN_IMSI1=010123456789
PRODUCT_M_IN_MSISDN1=BS11
PRODUCT_M_IN_VMS_NUMBER=358507770001
PRODUCT_M_OUT_PARAMS=
PRODUCT_NAME=Voice Only
PRODUCT_ACTIVITY_NAME=Create voice only subscription
PRODUCT_TRANSACTION_DURATION=300000
PRODUCT_TRANSACTION_NAME=Create
PRODUCT_TYPE=Product
PRODUCT_UNDO_TRANSACTION_NAME=Delete
PRODUCT_VERSION=2.0
```

Selection groups are evaluated in the same way as in the technical service level decomposition. Parameter references are generated as described in Section 3.6.3 *Parameter References*.

3.6.2.1 Get Only Mainline Parameters

If only mainline-related parameters are asked for, the step returns parameters related to mainline creation in the Gantt chart. The parameters are structured as described in Section 3.6.2 *Product Hierarchy Decomposition*, but the step will only contain parameters whose names end with:

- NAME
- VERSION
- TRANSACTION_NAME
- TRANSACTION_DURATION
- ITEM_TYPE
- ACTIVITY_NAME
- GROUP_COUNT
- GROUP_SIZE

All other parameters described in Section 3.6.2 *Product Hierarchy Decomposition* are filtered out.

3.6.3 Parameter References

In the Catalog model it is possible to map a response/complete parameter from a technical service /library to be used as the request parameter for a following technical service/library. BST Interface for Catalog supports this feature by giving the response parameter value as a parameter reference to the following technical service/library. For this mechanism to work, the technical services (tasks/libraries) need to add the task response parameters to Business Service Tool parameter pool with the name defined in the Catalog model.

Example:

The first technical service has a parameter `MSISDN` defined in the `Complete` message. The second technical service has a request parameter "`OLD_MSISDN`" which is mapped to use the MSISDN from the first technical service in the Catalog model. The `OLD_MSISDN` parameter for the second technical service is mapped in the following way: (`PRODUCT_` is used a prefix for decomposition):

```
PRODUCT_2_1_OLD_MSISDN=($MSISDN)
```

This mechanism also works when referencing parameters between technical services in different services. A limitation to this mechanism is, however, that the parameter names used for different references must not collide with each other.

3.6.4 Selection Groups

A product or service in Catalog may contain selection groups. Selection groups are used for selecting an item into the decomposition at runtime. A selection group contains a set of items and a rule for each item, defined in Catalog model. When decomposition is asked for a product or service that contains a selection group, BST Interface for Catalog evaluates the rules in the selection group and selects the first matching item into the decomposition. The evaluation is done using the selection group rules and the parameters in the BST parameter pool. If there are no matching rules, the item defined as the default in the selection group is selected.

Note A selection group may also contain an empty branch (no items defined for the branch).

Note If a parameter required for the evaluation is missing from the BST parameter pool, it does not cause a failure in the evaluation process. Instead the rule (or part of it) referring to a missing parameter is considered a "non-match". An exception to this is the `notSet` operator in the selection group.

Note A product or service may contain multiple (even nested) selection groups.

3.7 ErrorDecomposition Step

Use the `ErrorDecomposition` step to return rollback information from Catalog. This step uses the same search criteria as the `ServiceSpecification` step with additional `ERROR_DATA` information. This step functions in a similar way as the `ServiceDecomposition` step, but it returns a list of technical services that must be processed to undo all the changes made to the network so far.

The step requires a list of BST parameters as attributes and checks for the presence of mandatory parameters. If the parameter pool does not have the mandatory parameters needed for rollback, the step returns an error code `(SC_ERROR_CODE=1)`, but the logic execution is not terminated. The step creates the `SC_INVALID_ATTRIBUTE` parameter if there are invalid or missing parameters in the parameter pool when fetching error decomposition.

Error decomposition looks for rollback information first from the top level (requested) item. If an undo transaction has been defined for the top level item, error decomposition applies the **item level rollback** strategy, where:

- An item is returned for every executed technical service. Error data is used for determining whether the item was executed or not.

- All additional technical services from the top level undo transaction that were not present in the original decomposition are returned.
- These are grouped by the order in which they appear in the top level item undo transaction name.

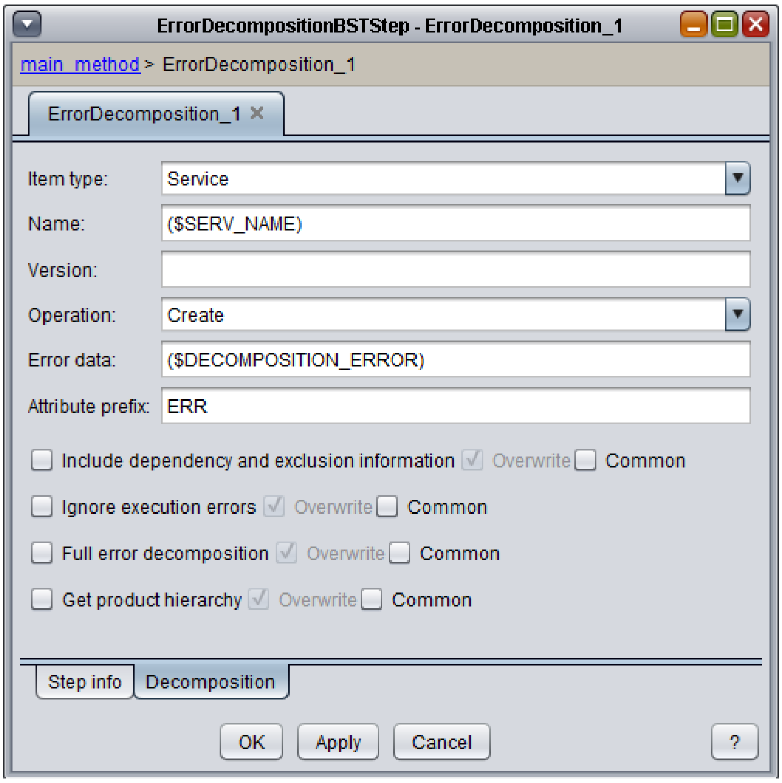
If the top level item does not have an undo transaction defined for it, error decomposition applies the **technical service level rollback** strategy, where:

- A rollback item is returned for every executed technical service/library (if their undo transaction has been defined). Error data is used for determining whether the item was executed or not.
- These are grouped by reversing the original execution order.

The `ErrorDecomposition` step can also be used to fetch the full error decomposition for a product or service if so configured. This means that all available rollback items are returned regardless of the error data value.

Optionally, parameters for full product hierarchy can be created in a similar way as with the `ServiceDecomposition` step. For more information, see Section 3.6.2 *Product Hierarchy Decomposition*.

Figure 7 illustrates the `ErrorDecomposition` step.



ErrorDecompositionBSTStep - ErrorDecomposition_1

main_method > ErrorDecomposition_1

Item type: Service

Name: ($SERV_NAME)

Version:

Operation: Create

Error data: ($DECOMPOSITION_ERROR)

Attribute prefix: ERR

☐ Include dependency and exclusion information ☑ Overwrite ☐ Common

☐ Ignore execution errors ☑ Overwrite ☐ Common

☐ Full error decomposition ☑ Overwrite ☐ Common

☐ Get product hierarchy ☑ Overwrite ☐ Common

Figure 7. `ErrorDecomposition` step

Table 9 lists the requested input parameters.

Table 9. `ErrorDecomposition` step input parameters

Parameter	UI field	Description	M/O
`SC_STEP_ENTITY_TYPE`	Item type	Item type. **Example values:** • `Product` • `Service` • `Technical Service`	`M`
`SC_STEP_ENTITY_NAME`	Name	Name of a product or service in Catalog, for example, the name of a product. **Value format:** `string` **Maximum length:** `80`	`M`
`SC_STEP_ENTITY_VERSION`	Version	Version of the item in Catalog. **Value format:** `<major version>.<minor version>` **Example value:** `1.0`	`O`
`SC_STEP_OPERATION`	Operation	Requested operation in Catalog. **Example values:** • `Create` • `Modify` • `Delete` • `Display`	`M`
`SC_STEP_ERROR_DATA`	Error data	`ERROR_DATA` of the technical service that has failed. The name of the parameter that, in case of error, is added to the parameter pool and contains information on the location of the failure. This parameter is not used when full error decomposition is requested.	`M/O`
`SC_STEP_PARAMETER_PREFIX`	Attribute prefix	Prefix for parameters including `ErrorDecomposition`.	`O`
`SC_STEP_DEPENDENCIES_EXCLUSIONS`	Include dependency and exclusion information	The step creates parameters for decomposition and technical service levels, for capabilities, dependencies and exclusions. There is also a parameter for missing dependencies on decomposition level. Each parameter contains a comma-separated list of values.	

Table 9. `ErrorDecomposition` step input parameters (Continued)

Parameter	UI field	Description	M/O
`SC_STEP_ENTITY_ID`		Integer number that uniquely identifies an item and its version. In Catalog UI the parameter is called External ID. The parameter does not have its own field in the **Specification** tab of the step but you can define it, for example, with the `MapParameters` step. If `SC_STEP_ENTITY_ID` is defined, also Search type (`SC_STEP_OPERATION`) must be defined. If both Name (`SC_STEP_ENTITY_NAME`) and `SC_STEP_ENTITY_ID` are defined, Name is used. **For example:** Product “Chat a lot” version 1.0 is identified with `SC_STEP_ENTITY_ID` 782 Product “Chat a lot” version 1.1 is identified with `SC_STEP_ENTITY_ID 1025` If `SC_STEP_ENTITY_ID` is defined, Search type (`SC_STEP_OPERATION`) must also be defined. However, Type (`SC_ENTITY_TYPE`), Name (`SC_STEP_ENTITY_NAME`) and Version (`SC_STEP_ENTITY_VERSION`) must be left empty.	O

Table 9. `ErrorDecomposition` step input parameters (Continued)

Parameter	UI field	Description	M/O
`SC_STEP_SERVICE_STATE`		State of a product or service. The parameter does not have its own field in the UI but you can define it, for example, with the `MapParameters` step. **Example values:** • `ALL` • `Testing` • `Suspended` • `Published` • `ANY` Value `ALL` returns the latest version that is in one of the states defined in the Catalog `all_lifecycle_states` parameter in the `client.proerties` file. For example, if the value of `all_lifecycle_states` is "Testing, Published, Suspended", the latest version that is in one of these states is returned.	O
`SC_IGNORE_ERRORS`	Ignore execution errors	If enabled, the `SC_ERROR_CODE` parameter is put to the parameter pool in case of an error. If disabled, BST logic execution stops. **Possible values:** • `TRUE` • `FALSE` (Default value)	O
`SC_STEP_FULL_ERROR_DECOMPOSITION`	Full error decomposition	Defines whether this step fetches the full error decomposition for the product or service. When selected, parameter `SC_STEP_ERROR_DATA` is ignored. **Possible values:** • `TRUE` • `FALSE` (Default value)	O
`SC_FULL_DECOMPOSITION`	Get product hierarchy	If enabled, the step creates parameters for full product hierarchy. For more information about the output parameters for full product hierarchy, see Section 3.6.2 *Product Hierarchy Decomposition*. **Possible values:** • `TRUE` • `FALSE` (Default value)	O

Table 10 lists the output parameters.

Table 10. `ErrorDecomposition` step output parameters

Parameter	Description

`<prefix><y>_<z>_<parameter name>=<parameter value>`	List of product/service attributes and values from Catalog and capability library.
	y=technical service group (task group)
	z=technical service (task)
	The technical service group number defines the order in which the tasks must be sent to InstantLink. If there are several tasks within a group, they can be sent using the `ParallelSend2` BST step.
	Example value:
	`NEW_1_1_NE_TYPE=HLR`

`<prefix>CAPABILITIES`	List of capabilities on decomposition level. **Example value:** `NEW_CAPABILITES=ADSL, BASIC_GSM, VMS, SMS, BLACKBERRY`
`<prefix>DEPENDENCIES`	List of dependencies on decomposition level. **Example value:** `NEW_DEPENDENCIES=INTERNET_ACCESS, MOBILE_VOICE`
`<prefix>EXCLUSIONS`	List of exclusions on decomposition level. **Example value:** `NEW_1_1_CAPABILITIES=ADSL`
`<prefix>MISSING_DEPENDENCIES`	List of missing dependencies on decomposition level. **Example value:** `NEW_MISSING_DEPENDENCIES=MOBILE_VOICE`

`<prefix><y>_<z>_CAPABILITIES`	List of capabilities on lowest/technical service level.
	y=technical service group (task group)
	z=technical service (task)
	Example value:
	`TS_ADSL,TS_GRPS,TS_INVENTORY`

`<prefix><y>_<z>_DEPENDENCIES`	List of dependencies on lowest/technical service level.
	y=technical service group (task group)
	z=technical service (task)
	Example value:
	`NEW_1_1_DEPENDENCIES= INTERNET_ACCESS`

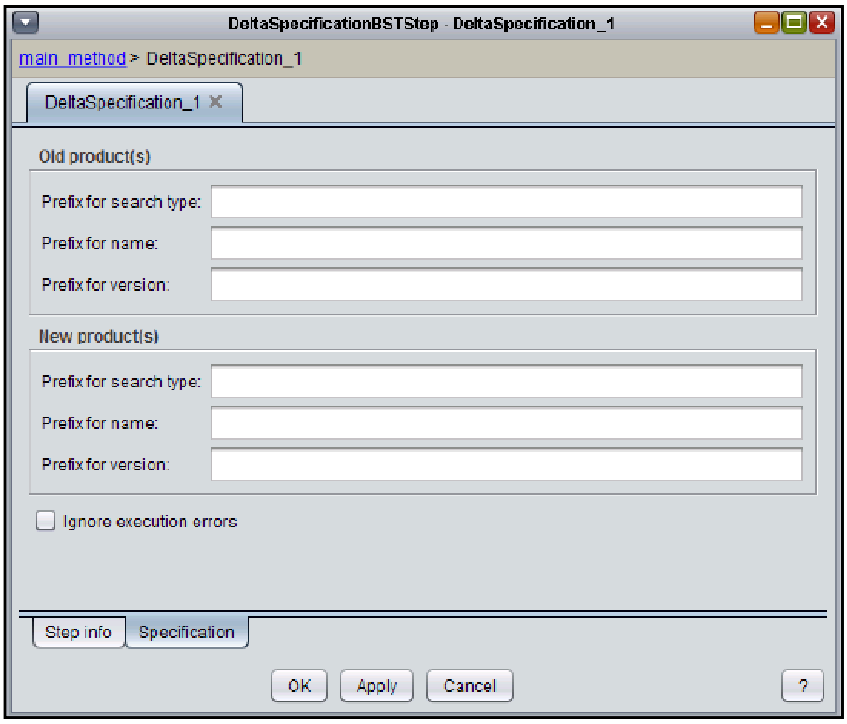
`<prefix><y>_<z>_EXCLUSIONS`	List of exclusions on lowest/technical service level.
	y=technical service group (task group)
	z=technical service (task)
	Example value:
	`NEW_1_1_EXCLUSIONS=VDSL`

Table 10. `ErrorDecomposition` step output parameters (Continued)

Parameter	Description
`SC_GROUP_COUNT`	Number of technical service groups in the parameter pool. **Example value:** `SC_GROUP_COUNT=1`
`SC_GROUP_SIZE_<X>`	Number of technical services within each group. x=Technical service group identifier **Example value:** `SC_GROUP_SIZE_1=6`

3.8 DeltaSpecification Step

Use the `DeltaSpecification` step to return the specification information from Catalog when migrating from one product or service to another. The returned list of parameters (mandatory and optional) for the provisioning task is based on the delta calculation. This step must be executed before the `DeltaCalculation` step. *Figure 8* illustrates the `DeltaSpecification` step.



DeltaSpecificationBSTStep - DeltaSpecification_1

main_method > DeltaSpecification_1

DeltaSpecification_1

Old product(s)

Prefix for search type:

Prefix for name:

Prefix for version:

New product(s)

Prefix for search type:

Prefix for name:

Prefix for version:

☐ Ignore execution errors

Figure 8. `DeltaSpecification` step

Table 11 lists the `DeltaSpecification` step parameters that define the prefixes for parameters which collect existing and new product/service information for the delta calculation.

Table 11. `DeltaSpecification` step parameters

Parameter name	UI field	Description	M/O
`SC_DELTA_PREFIX_TYPE_OLD`	Prefix for search type	Prefix for parameters (in the parameter pool) containing the search type (Product, Service or Technical Service) to migrate from. **Example:** `OLD_PRODUCT_TYPE_` locates parameters starting with `OLD_PRODUCT_TYPE_` , for example, `OLD_PRODUCT_TYPE_1`	`M`
`SC_DELTA_PREFIX_NAME_OLD`	Prefix for name	Prefix for parameters containing the name of the Product, Service or Technical Service to migrate from. **Example:** `OLD_PRODUCT_NAME_`	`M`
`SC_DELTA_PREFIX_VERSION_OLD`	Prefix for version	Prefix for parameters containing the version of the Product, Service or Technical Service to migrate from. **Example:** `OLD_PRODUCT_VERSION_`	`O`
`SC_DELTA_PREFIX_TYPE_NEW`	Prefix for search type	Prefix for parameters containing the search type (Product, Service or Technical Service) to migrate to. **Example:** `NEW_PRODUCT_TYPE_` locates parameters starting with `NEW_PRODUCT_TYPE_` , for example, `NEW_PRODUCT_TYPE_1`	`M`
`SC_DELTA_PREFIX_NAME_NEW`	Prefix for name	Prefix for parameters containing the version of the Product, Service or Technical Service to migrate to. **Example:** `NEW_PRODUCT_VERSION_`	`M`
`SC_DELTA_PREFIX_VERSION_NEW`	Prefix for version	Prefix for parameters containing the version of the Product, Service or Technical Service to migrate to. **Example:** `NEW_PRODUCT_VERSION_`	`O`

Table 11. `DeltaSpecification` step parameters (Continued)

Parameter name	UI field	Description	M/O
`SC_STEP_SERVICE_STATE`		The state that the product, service or technical service must be in (in Catalog) to be considered for delta calculation. The parameter does not have its own field in the UI but you can define it, for example, with the `MapParameters` step. **Example values:** • `ALL` • `Testing` • `Suspended` • `Published` • `ANY` Value `ALL` returns the latest version that is in one of the states defined in the Catalog `all_lifecycle_states` parameter in the `client.proerties` file. For example, if the value of `all_lifecycle_states` is "Testing, Published, Suspended", the latest version that is in one of these states is returned.	O
`SC_DELTA_PREFIX_ID_OLD`		Prefix for parameters containing External IDs that identify the old products and their version. External ID is an integer number that uniquely identifies and item and its version. The parameter can be used instead of `SC_DELTA_PREFIX_NAME_OLD` (Prefix for name). If both are provided, `SC_DELTA_PREFIX_NAME_OLD` is used. The parameter does not have its own field in the UI, but you can define it, for example, with the `MapParameters` step.	O
`SC_DELTA_PREFIX_ID_NEW`		Prefix for parameters containing External IDs that identify the new products and their version. External ID is an integer number that uniquely identifies and item and its version. The parameter can be used instead of `SC_DELTA_PREFIX_NAME_NEW` (Prefix for name). If both are provided, `SC_DELTA_PREFIX_NAME_NEW` is used. The parameter does not have its own field in the UI, but you can define it, for example, with the `MapParameters` step.	O

Table 11. `DeltaSpecification` step parameters (Continued)

Parameter name	UI field	Description	M/O
`SC_STEP_OPERATION`		Requested operation in Catalog. **Default value:** • `Create` **Example values:** • `Create` • `Modify` • `Delete` • `Display` The parameter does not have its own field in the UI, but you can define it, for example, with the `MapParameters` step.	O
`SC_IGNORE_ERRORS`	Ignore execution errors	If enabled, `SC_ERROR_CODE` parameter is put to the parameter pool in case of an error. If disabled, BST logic execution stops. **Possible values:** • `TRUE` • `FALSE` (Default value)	O
`SC_STEP_DEPENDENCIES_EXCLUSIONS`		Define whether you want Business Service Tool Engine to create parameters for capabilities, dependencies and exclusions. **Possible values:** • `TRUE` • `FALSE` (Default value) The parameter does not have its own field in the UI, but you can define it, for example, with the `MapParameters` step.	O

Example: Delta calculation with product names

A subscription currently has products Mobile Voice version 1.0 and Mobile Data version 1.0. The subscription is going to be migrated to products Mobile Voice 2.1 and Mobile Data 3.0. The parameter pool has the following existing product/service information defining the current products:

```
OLD_PRODUCT_NAME_1=Mobile Voice
OLD_PRODUCT_TYPE_1=Product
OLD_PRODUCT_VERSION_1=1.0
OLD_PRODUCT_NAME_2=Mobile Data
OLD_PRODUCT_TYPE_2=Product
OLD_PRODUCT_VERSION_2=1.0
```

The parameter pool has the following information of the new products:

```
NEW_PRODUCT_NAME_1=Mobile Voice
NEW_PRODUCT_TYPE_1=Product
NEW_PRODUCT_VERSION_1=2.1
NEW_PRODUCT_NAME_2=Mobile Data
NEW_PRODUCT_TYPE_2=Product
NEW_PRODUCT_VERSION_2=3.0
```

The `DeltaCalculation` step defines the following parameters and returns parameters from the parameter pool according to the given prefixes:

```
SC_DELTA_PREFIX_NAME_OLD=OLD_PRODUCT_NAME_
SC_DELTA_PREFIX_TYPE_OLD=OLD_PRODUCT_TYPE_
SC_DELTA_PREFIX_VERSION_OLD=OLD_PRODUCT_VERSION_
SC_DELTA_PREFIX_NAME_NEW=NEW_PRODUCT_NAME_
SC_DELTA_PREFIX_TYPE_NEW=NEW_PRODUCT_TYPE_
SC_DELTA_PREFIX_VERSION_NEW=NEW_PRODUCT_VERSION_
```

Example: Delta calculation with External IDs

Delta calculation can be done based on External IDs that uniquely identify an item and its version. Even though it is supported, it is recommended to use product names instead.

The parameter pool has the following existing product/service information defining the current products:

```
OLD_PRODUCT_ID_1=403
OLD_PRODUCT_TYPE_1=Product
OLD_PRODUCT_ID_2=584
OLD_PRODUCT_TYPE_2=Product
```

The parameter pool has the following information of the new products:

```
NEW_PRODUCT_ID_1=642
NEW_PRODUCT_TYPE_1=Product
NEW_PRODUCT_ID_2=677
NEW_PRODUCT_TYPE_2=Product
```

The `DeltaCalculation` step defines the following parameters and returns parameters from the parameter pool according to the given prefixes:

```
SC_DELTA_PREFIX_ID_OLD=OLD_PRODUCT_ID_
SC_DELTA_PREFIX_TYPE_OLD=OLD_PRODUCT_TYPE_
SC_DELTA_PREFIX_ID_NEW=NEW_PRODUCT_ID_
SC_DELTA_PREFIX_TYPE_NEW=NEW_PRODUCT_TYPE_
```

Table 12 lists the output parameters.

Table 12. Output parameters

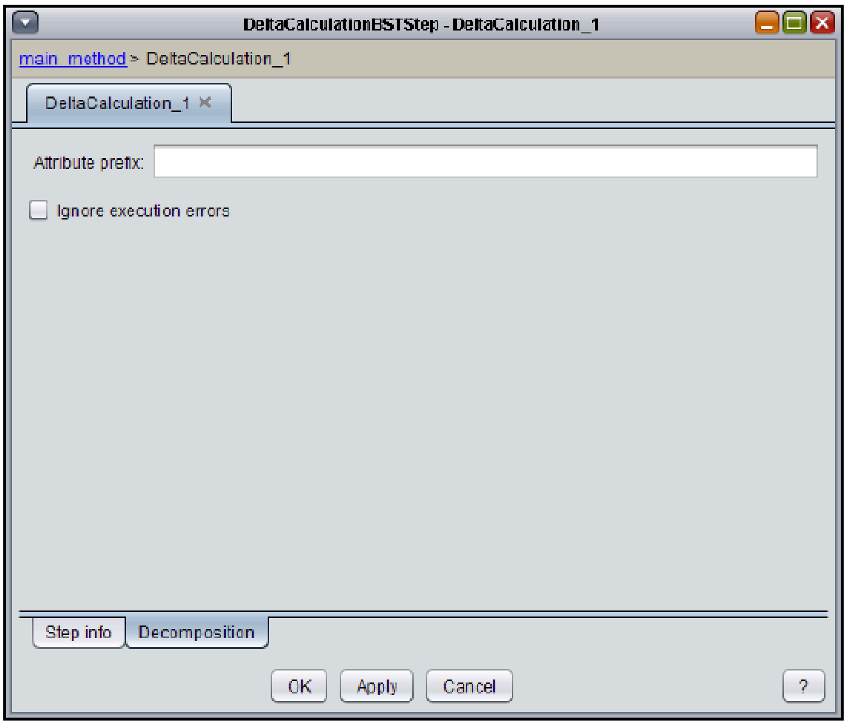
Parameter name	Description
SC_MISSING_OPTIONAL	List of optional parameters.
SC_MISSING_MANDATORY	List of mandatory parameters.

The `DeltaSpecification` step fails if an item that should be rolled back has no undo transaction defined for it. If `SC_IGNORE_ERRORS` has been defined as `FALSE` (default), the logic execution stops with an appropriate error message. If `SC_IGNORE_ERRORS` has been defined as `TRUE`, the following parameters are added to the parameter pool and the logic execution continues:

Parameter name	Description
SC_ERROR_CODE - **903**	Error code value `903` indicates that the undo transaction has not been defined for a product, service or technical service
SC_ MISSING_UNDO_ITEM_NAME	List of products/services/technical services that do not have undo transaction defined for them.
SC_ MISSING_UNDO_ITEM_VERSION	List of product,/service/technical service versions that do not have undo transaction defined for them.

3.9 DeltaCalculation Step

Use the `DeltaCalculation` step calc to return the decomposition of a product/service upgrade/downgrade. Perform this step after the `DeltaSpecification` step. *Figure 9* illustrates the `DeltaCalculation` step.


DeltaCalculationBSTStep - DeltaCalculation_1
main_method > DeltaCalculation_1
DeltaCalculation_1 ×
Step info
Decomposition
OK
Apply
Cancel
?

Attribute prefix:

□ Ignore execution errors

Figure 9. `DeltaCalculation` step

Table 13 lists the input parameters.

Table 13. `DeltaCalculation` step input parameters

Parameter name	UI field	Description	M/O
`SC_STEP_PARAMETER_PREFIX`	Attribute prefix	Prefix that is added to all the parameters that are returned by the delta calculation. The prefix allows identifying which parameters in the parameter pool result from delta calculation done by Catalog.	O
`SC_IGNORE_ERRORS`	Ignore execution errors	If enabled, `SC_ERROR_CODE` parameter is put to the parameter pool in case of an error. If disabled, BST logic execution stops. **Possible values:** • `TRUE` • `FALSE` (Default value)	O

A number of prefixed parameters that belong to a technical service are returned to the step. That to change a subscriber's subscription from the old product to the new product.

Table 14 lists the output parameters.

Table 14. `DeltaCalculation` step output parameters

Parameter	Description

`<prefix><y>_<z>_<parameter name>=<parameter value>`	List of attributes and values from Catalog.
	y=technical service group (task group)
	z=technical service (task)
	Technical service group number defines the order in which the tasks must be sent to InstantLink. **Example value:**
	`CAPABILITY_1_1_NE_TYPE=RATER`

`<prefix>CAPABILITIES`	List of capabilities on decomposition level. **Example value:** `NEW_CAPABILITES=ADSL, BASIC_GSM, VMS, SMS, BLACKBERRY`
`<prefix>DEPENDENCIES`	List of dependencies on decomposition level. **Example value:** `NEW_DEPENDENCIES=INTERNET_ACCESS, MOBILE_VOICE`
`<prefix>EXCLUSIONS`	List of exclusions on decomposition level. **Example value:** `NEW_1_1_CAPABILITIES=ADSL`
`<prefix>MISSING_DEPENDENCIES`	List of missing dependencies on decomposition level. **Example value:** `NEW_MISSING_DEPENDENCIES=MOBILE_VOICE`

`<prefix><y>_<z>_CAPABILITIES`	List of capabilities on lowest/technical service level.
	y=technical service group (task group)
	z=technical service (task)
	Example value:
	`TS_ADSL,TS_GRPS,TS_INVENTORY`

`<prefix><y>_<z>_DEPENDENCIES`	List of dependencies on lowest/technicalservice level.
	y=technical service group (task group)
	z=technical service (task)
	Example value:
	`NEW_1_1_DEPENDENCIES= INTERNET_ACCESS`

`<prefix><y>_<z>_EXCLUSIONS`	List of exclusions on lowest/technical service level.
	y=technical service group (task group)
	z=technical service (task)
	Example value:
	`NEW_1_1_EXCLUSIONS=VDSL`

Table 14. `DeltaCalculation` step output parameters (Continued)

Parameter	Description
`<prefix><y>_<z>_TS_NAME`	Name of the lowest level item (technical service) in Catalog. **Example value:** `HLR_voice` If several lowest level items are grouped in Catalog, the group is passed as one item to BST and `TS_NAME` contains the name of the group.
`<prefix><y>_<z>_TS_VERSION`	Version of the lowest level item (technical service) in Catalog **Example value:** `4.1`
`<prefix><y>_<z>_TS_TYPE`	Type of the lowest level item (technical service) in Catalog. **Example value:** `TS`
`SC_GROUP_COUNT`	Number of technical service groups in the parameter pool. **Example value:** `SC_GROUP_COUNT=2`

`SC_GROUP_SIZE_<X>`	Number of technical services within each group.
	x=Technical service group identifier
	Example value:
	`SC_GROUP_SIZE_1=2`
	`SC_GROUP_SIZE_2=4`

Examples:

The old and the new products are almost identical. However, there is one difference: the old product has basic service BS26 whereas the new product has service basic BS28. The Delta calculation step returns two tasks:

Undo tasks to remove a BS26 service:

```
DC_1_1_REQ_TYPE 3
DC_1_1_REQ_OBJ 2
DC_1_1_NE_TYPE HLR
DC_1_1_BASIC1 BS26
DC_1_1_IMSI1 3434343434343
```

Create a task to add a BS28 service:

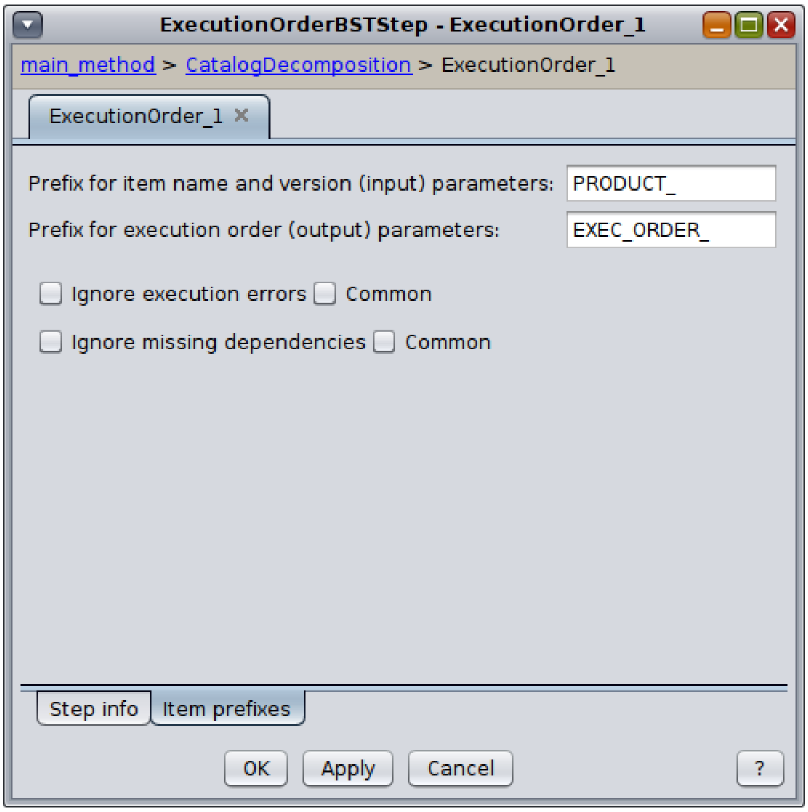
```
DC_2_1_REQ_TYPE 1
DC_2_1_REQ_OBJ 2
DC_2_1_NE_TYPE HLR
DC_1_1_BASIC1 BS28
DC_1_2_IMSI1 3434343434343
```

3.10 ExecutionOrder Step

The `ExecutionOrder` step can be used to fetch the execution order for given Catalog items. The execution order of a set of items is queried by providing a list of said items to Catalog, which then calculates the correct processing order using its knowledge of capabilities and dependencies of the given items, and provides the correct processing order as a response. The list of items used in the query is generated from the parameter pool using a specific item prefix and parameter naming convention. As a result of the query, the execution order of items is returned into the parameter pool also using a specific item prefix and parameter naming convention.

Item version and lifecycle state can be defined in addition to item name for fetching the execution order. They can be used, for example, for selecting a particular version of the item or the latest item in a specific state for the query. If version and lifecycle state are not defined, the query uses an item that is in state defined by `all_lifecycle_states` with the highest version number.

Figure 10 illustrates the `ExecutionOrder` step.



ExecutionOrderBSTStep - ExecutionOrder_1

main_method > CatalogDecomposition > ExecutionOrder_1

Prefix for item name and version (input) parameters: PRODUCT_

Prefix for execution order (output) parameters: EXEC_ORDER_

□ Ignore execution errors □ Common

□ Ignore missing dependencies □ Common

Figure 10. `ExecutionOrder` step

Table 15 lists the requested input parameters.

Table 15. `ExecutionOrder` step input parameters

Parameter	UI field	Description	M/O

`SC_STEP_ITEM_PREFIX`	Prefix for item name, lifecycle state and version (input) parameters	Specifies the prefix that is used for looking up the item information from the parameter pool.
		If no value is given, the default value `ITEM_` is used.
		The following parameters are searched for from the parameter pool:
		`<prefix><1..n>_ITEM_NAME`
		`<prefix><1..n>_ITEM_STATE`
		`<prefix><1..n>_ITEM_VERSION`
		For each found item name, lifecycle state and version, there is an entry in the list that is used for fetching the execution order.
		If the item state is not given, value `ALL` is used by default. States defined in `all_lifecycle_states` are then considered valid.
		If the item state value is `ANY`, any state for an item is considered valid.
			O

`SC_STEP_EXEC_ORDER_PREFIX`	Prefix for execution order (output) parameters	Defines the prefix that is used for returning the execution order into the parameter pool. If no value is given, the default value `EXEC_ORDER_` is used.	O
`SC_IGNORE_ERRORS`	Ignore execution errors	If enabled, `SC_ERROR_CODE` parameter is put to the parameter pool in case of an error. If disabled, BST logic execution stops. **Possible values:** • `TRUE` • `FALSE` (Default value)	O
`SC_STEP_EXEC_ORDER_IGNORE_MISSING_DEPENDENCIES`	Ignore missing dependencies	By default, execution order query fails in Catalog if all dependent items are not part of the query. When this option is enabled, Catalog returns an execution order even if all dependencies are not fulfilled. **Possible values:** • `TRUE` • `FALSE` (Default value)	O

The result of the execution order query contains the items grouped according to their dependencies. Items that can be executed in parallel reside in the same group. The information on grouped items follows the syntax described in *Table 16*.

Table 16. `ExecutionOrder` step output parameters

Parameter	Description

`<prefix><x>_<y>_ITEM_NAME`	The name of the item.
	x = group
	y = index
	Example value:
	`EXEC_ORDER_1_1_ITEM_NAME P1`

`<prefix><x>_<y>_ITEM_VERSION`	The version of the item.
	x = group
	y = index
	Example value:
	`EXEC_ORDER_1_1_ITEM_VERSION 1.0`

`<prefix><x>_<y>_INDEX`	The index of the item that was used in the query.
	x = group
	y = index
	Example value:
	`EXEC_ORDER_1_1_INDEX 4`

Example:

Figure 11 shows an example of the hierarchy of four products as defined in Catalog.

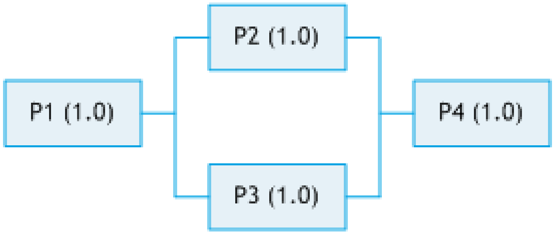


Figure 11. Product hierarchy example

Input parameters for the execution order query (with input prefix `PRODUCT_`):

```
PRODUCT_1_ITEM_NAME P3
PRODUCT_1_ITEM_STATE Published
PRODUCT_1_ITEM_VERSION 1.0
PRODUCT_2_ITEM_NAME P2
PRODUCT_2_ITEM_STATE Published
PRODUCT_2_ITEM_VERSION 1.0
PRODUCT_3_ITEM_NAME P4
```

```
PRODUCT_3_ITEM_STATE Published
PRODUCT_3_ITEM_VERSION 1.0
PRODUCT_4_ITEM_NAME P1
PRODUCT_4_ITEM_STATE Published
PRODUCT_4_ITEM_VERSION 1.0
```

The following execution order is added to the parameter pool (with output prefix EXEC_ORDER_) after running the query. There are three groups of items where the first and third group have one item and the second group has two parallel items.

```
EXEC_ORDER_1_1_ITEM_NAME P1
EXEC_ORDER_1_1_ITEM_VERSION 1.0
EXEC_ORDER_1_1_INDEX 4
EXEC_ORDER_2_1_ITEM_NAME P3
EXEC_ORDER_2_1_ITEM_VERSION 1.0
EXEC_ORDER_2_1_INDEX 1
EXEC_ORDER_2_2_ITEM_NAME P2
EXEC_ORDER_2_2_ITEM_VERSION 1.0
EXEC_ORDER_2_2_INDEX 2
EXEC_ORDER_3_1_ITEM_NAME P4
EXEC_ORDER_3_1_ITEM_VERSION 1.0
EXEC_ORDER_3_1_INDEX 3
```

3.11 Usage of Logic Library Logics

The `LogicLibrary` step in Business Service Tool allows the calling of library logics. Library logics are common logics that are used by other provisioning logics. To help with the handling of complex cases within a logic that uses Catalog, you can define the called logic library logics as a technical library attribute value in Catalog.

The `LogicLibrary` step in a logic can be used to dynamically call the library logic with a name defined as a parameter value.

3.12 Error Codes

If the step execution fails and ignore execution errors check box is checked (`SC_IGNORE_ERRORS=TRUE`), the reason for the failure is added to the parameter pool in parameter `SC_ERROR_CODE` and the BST logic execution continues. Step parameter `SC_ERROR_MESSAGE` has an error message related to the error.

`SC_ERROR_CODE` can have one of the following values:

SC_ERROR_CODE	Description
1	The parameter pool does not have the mandatory parameters needed for the rollback (in `ErrorDecomposition` step).
600	A general Catalog error.
900	A Catalog connection error.
901	A general Catalog exception.
902	An item is not available in Catalog.

SC_ERROR_CODE	Description
`903`	Undo transaction is not defined for the item (in `DeltaSpecification` step).
`904`	Items have missing dependencies.
`905`	An item is available in Catalog, but it does not contain the transaction in question.