NW-ESR-CNT 1.0

Test Plan for
ESR-Content Certification

(Version 1.0)

Copyright

Trademarks

SAP and ABAP are registered trademarks of SAP AG. All other products are either registered or non-registered trademarks of the respective companies.

1 Introduction

Enterprise Services Repository (ESR) Content Certification intends to verify the conformance of ESR Content developed by Software Solution Providers to the guidelines set by SAP.

The certification focuses on modeling and defining A2X enterprise services (ES). A2X enterprise services are web services for synchronous exchange of business information of an application with an unspecified client (i.e. UI or Composite) with the focus on transmission of all information relevant for understanding the message. A2A (application-to-application) and B2B (business-to-business) Enterprise Services (synchronous or asynchronous) are not in the scope of the certification.

The scope of the certification also involves the verification of the implemented functionality of the ES to be certified.

2 ESR-Content Certification – General Requirements and Overview

2.1 Enterprise Service Provisioning License

Every service interface that is subject to the ESR content certification must have been licensed according to the SAP enterprise service provisioning license.

2.2 Development

2.2.1 Tool information

Developers use the Enterprise Services Repository (ES Repository) to model and specify objects that are to be implemented later at various levels of detail. You use the Enterprise Services Builder (ES Builder) to access the objects in the ES Repository.

Please refer to the SAP online help for more information on [1] Enterprise Services Repository for SAP NetWeaver CE.

The Enterprise Services Repository is part of the following SAP products:

- SAP NetWeaver Composition Environment 7.1
- SAP NetWeaver Process Integration 7.1

Software Solution Providers can obtain associated development license from SAP or access a hosted version of Enterprise Services Repository as part of SAP Integration and Certification Center's (ICC) Remote Access Connectivity (RAC) service.

2.2.2 Software Landscape Directory (SLD)

Within the SLD the vendor creates its own products and software components as a basis for the development of ESR-Content. A Software Component groups together all development objects in the Enterprise Services Repository and is the basis for the versioning of the content-model. A Software Component can be used in several products. More information on SLD can be found here [2] .

Software solution providers are required to follow the naming conventions described in the *Enterprise Service Technical Requirements for Certification* document [4].

2.2.3 Overview of certification-relevant content

The ESR content certification only certifies ESR artifacts that are required to create synchronous A2X service interfaces. These artifacts can be grouped into two categories:

- Modeling Content:
  All required artifacts are modeled within Process Component Models.
- Definition Content:
  The service definition artifacts target at deriving the actual service interface fine structure which in the end results in aWSDL.

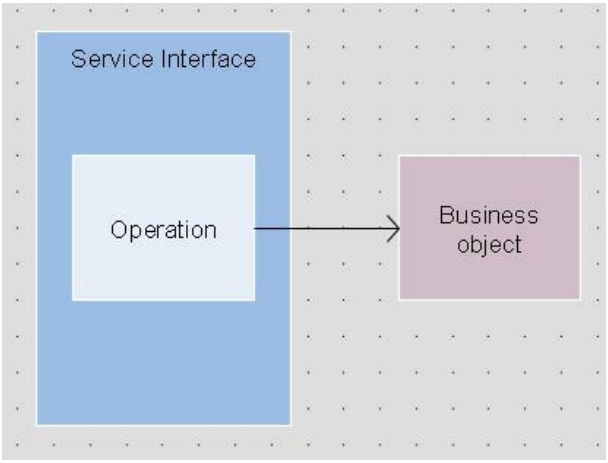
2.2.3.1 Overview of Model Types [3]

This certification only verifies the content of:

- **Process Component Model (SAP ProComp model)**: This model is used to describe the collection of service functionality exposed by a process component. It shows the business objects representing the data and service interfaces and operations that are used to access this data.

  The Process Component Model name must be equal to the process component's name. (*Mandatory*)

  The following picture shows different elements of a Process Component Model:



Process Component	Process components logically group business objects (BOs). A BO belongs to exactly one process component. Process components describe the part of a value chain that is normally (in large companies) performed in a department.
Business Object	A business object represents a specific view of data of a well-defined business area. Business objects are defined in such a way that they do not overlap and thus create business functionality redundancies.
Service Interface	A service interface groups operations. There is a limited set of service interfaces per business object. Operations are assigned to one of these service interfaces via the *interface patterns*.
Operation	An operation is a method of a business object. A business object can have more than one operation.

I.e. that per Process Component one Process Component Model is created with its name equal to the process component's name. This model itself then contains business objects, service interfaces, and operations.

The test plan assumes that software solution providers will only have one process component model as their ESR content deliverable. If however, there are multiple process components delivered by the software solution provider the requirements should be applied to each process component model separately.

2.2.3.2 Overview of service definition objects

This certification verifies the following artifacts:

Repository Namespace	A repository namespace is used to group all other definition artifacts to ensure that no naming conflicts occur. By default, these namespaces are used in the service interface WSDL.
Service Interface	A service interface represents the actual service. It generally contains one or more operations. From the service interface, the WSDL is derived.
Operation	An operation exposes individual service functionality. As only synchronous services are certificiable its structure is defined via its request, response and fault message types.
Message Type	The message types represent the SOAP-compliant messages that are exchanged during service communication (synchronous: request, response, fault).
Fault Message Type	The fault message type is always set to a SAP pre-defined type called StandardMessageFault.
Business Data Type	The fine structure of individual message types is set up by using data types.

2.2.4 Implementation

Implementation of the enterprise services delivered by the software solution provider is generally not in the scope of ESR-Content certification.

Nevertheless, there are some very general SOA implementation requirements that must be fulfilled. These are described in chapter 3.6 Implementation Requirements.

The following certifications are highly recommended where applicable:

- SAP J2EE Deployment certification.
- SAP ABAP certification.

2.2.5 Integration with SAP solutions

If the implementation of the enterprise services delivered by the software solution provider integrates with one or more SAP solutions (SAP NetWeaver, ERP, CRM, SCM, PLM, etc.), associated SAP interface should be certified. This is a pre-requisite for the ESR-content certification.

2.3 Documentation

2.3.1 Documentation concept

To maximize the value of ESR content, the technical solution must be supplemented by comprehensive documentation explaining the most efficient way to use the ESR content.

This documentation comprises three areas:

- Documentation of the Process Component Model and its components
- Consumption Document for each Enterprise Service Operation
- Installation and Configuration Guide

For certification reasons, all documentation should be provided at least in English.

2.3.2 Documentation of the Process Component Model

The certification requires that the following objects are documented:

- Process Components
- Business Objects
- Service Interfaces
- Operations

It is generally recommended that all description/definition fields for ESR content objects are maintained to maximize their semantic value.

Note: This document requires certain ones of these to be maintained!

There is a template available named ESR-CNT_Documentation_Template.doc that must be adhered to.

If ESR objects from Enterprise Services Workplace are re-used, the documentation must be copied from there!

It is generally recommended to refer to the corresponding documentation from the Enterprise Services Workplace as a guideline for documenting your own ESR content objects.

2.3.3 Consumption Document for each Enterprise Service Operation

Each enterprise service operation should be documented including a sample SOAP request and response.

Documentation should at least cover the following:

- Meaning of each message parameter
- Parameter values and/or value lists (where applicable)
- Parameter dependencies, integrity conditions (where applicable)
- Error situations and related messages (functional, returned using the Log datatype) that can be returned
- Exception situations and related messages (technical, returned using the StandardDefaultMessage datatype) that can be returned

For services supporting multiple languages, documentation should be provided at least in English.

2.3.4 Installation and Configuration Guide

The installation and configuration guide contains a step-by-step description (preferably with screenshots) of the installation and the configuration process for both the ESR content and the implementation of the service interfaces.

2.4 Deployment

ESR content is deployed using the import/export facility in the ES Builder. For more information about this transport mechanism see the online help [4].

2.5 Functionality Testing

You must demonstrate that the services work as expected. To do this, list the necessary functional test cases in the associated section of the Technical Product Profile. Specify enough test cases to check the functional correctness of all service operations provided with your content.

The ICC consultant will review and recommend changes on your suggested test cases as needed. As a result, functionality testing will be done based on a mutually agreed test plan.

2.6 Support and Solution Manager Readyness

Software Solution Providers must provide support for their enterprise services.

Support information (e.g. telephone, email, website, etc.) will be noted in the certification test report.

SAP now collects all the necessary meta-data to create an entry for certified Third Party Vendor Solutions in the Product and Production Management System (PPMS). PPMS is the central knowledge repository for technical information regarding SAP software products and their software components. SAP now utilizes PPMS to collect Non-SAP software products and their software components too.

Please check the documents “Step by Step Guide for Solution Manager Ready and PPMS Data Collection.pdf” and “Solution Manager Ready and PPMS Data Collection.pdf” for further details.

The data collection sheet is easy to fill out. Simply follow Step by Step Guide for Solution Manager Ready document and fill in the Product Name/Version fields and Software Component Name/Version fields of the PPMS sheet with exactly the same values entered in the SLD (see naming convention on page 5). Please continue to fill out the rest of the fields as required of the PPMS sheet.

After SAP had entered your PPMS data into the central knowledge repository system, all SAP customers will receive your Product and Software Component information through the regular SLD content download.

3 ESR-Content Certification – Detailed Requirements

3.1 System Landscape Information Requirements

The following naming conventions for SLD-objects must be adhered to:

SLD-Entry	Naming-convention	Example
Product Vendor	<Vendor Domain Name>	sap.com
Product Name	<Area>	ERP, CRM,… or Plant Maintenance, Order Management, …
Product Version	<Content-Version>	1.0
Software Component Name	[<Vendor Name_>]<Software Component>	SAP_HR
Software Component Caption	[<Vendor Name>] <Software Component Description>	SAP HR
Software Component Version	<Software Component Version>	1.0

3.2 Structural Enterprise Services Requirements

3.2.1 Software Component Version and Implementation Platform

If the implementation platform of the service interfaces described by the certification platform is SAP NetWeaver 7.0 ABAP [e.g. SAP ECC 6.0] or any other platform that has the restriction that per web service interface there can be only one operation, the attribute *Use of Interface Objects* of the Software Component Version containing the service interface model and definition must be set to “SAP NetWeaver 7.0”.

In any other case the attribute *Use of Interface Objects* of the Software Component Version containing the service interface model and definition must be set to “SAP NetWeaver 7.1”.

Note: NW 7.1 SWCVs are not downward compatible and cannot be imported in the NW PI 7.0 interface repository.

3.2.2 Namespaces

The following naming convention for repository namespaces must be adhered to:

Naming convention	Example
http://<Vendor Domain Name>/<sub-domain name>	http://sap.com/erp

3.3 General Naming Requirements

3.3.1 Abbreviations

Generally, abbreviations in names should be avoided, but if an abbreviation is required, the following abbreviation rules for names must be adhered to:

Names using Title Case:

Take each separate **word** of the name and abbreviate it according to the code list of GDT *AbbreviationCode* in the Global Data Types catalog on SDN. If no abbreviation is available define your own abbreviations, if possible based on common abbreviations.

Example: Sales Order Management => Sls Ord Mgmt

Names using CamelCase:

Take each **qualifier** of the name and abbreviate it according to the code list of GDT *AbbreviationCode* in the Global Data Types catalog on SDN. If no abbreviation is available define your own abbreviations, if possible based on common abbreviations.

Example:
SalesOrderCreateRequest_syncItem = Sales+Order+Create+Request+_sync+Item =>
Sls+Ord+Crte+Req+_sync+Itm= SlsOrdCrteReq_syncItm

3.4 Modeling Content Requirements

3.4.1 General harmonization requirements

Each BO contained in the certification content must be either already existing in the Enterprise Services Workplace (a snapshot of these already existing BOs and their definitions can be found in Exhibit B of the Enterprise Services Provisioning License - Business Object Definitions) or it must be defined according to the rules stated in the following sub-chapter.

If applicable, existing BOs must be re-used which in turn means that if own BOs are defined they must be semantically clearly distinct from and not overlapping with existing SAP or previously defined own existing BOs.

Example: If the solution is a procurement fulfillment application your central BO should not be a Procurement Order but a Purchase Order. Both terms are actually equivalent from a language perspective but the BO Purchase Order is pre-defined in the SAP Enterprise Services Workplace and thus must be re-used.

The vendor affirms that this requirement has been fulfilled to best knowledge.

If the certification content re-uses a BO which can be found in the Enterprise Services Workplace the process component containing the BO must be re-used in the certification content as well.

Example: The ESR content shall model and define service operations around the business object Sales Order. As this business object is part of the process component Sales Order Processing, the ESR content to be certified must contain a process component model named Sales Order Processing, along with the corresponding business object Sales Order.

If the certification content re-uses a BO and/or process component which can be found in the Enterprise Services Workplace, all related descriptions and definitions for the model objects representing these BOs and/or process components must be re-used in the certification content as well.

3.4.2 Process components

In the following lists of process component aspects only the ones marked fat are subject to certification. All other aspects are required but will not be explicitly checked during certification. The vendor affirms that these requirements have been fulfilled to best knowledge.

A process component must describe a self-contained part of the value chain, typically executed by one department.

Every process component's name must

(a) be unique at least within your own business solution domain;
(b) describe the executed process aspect
(c) be in British English
(d) use gerund end forms, if possible (Most common SAP end forms: Processing, Management, Administration; prominent example: Purchase Order Processing)
(e) have qualifiers always precede the name qualified;
(f) be in Title Case
(g) be, as a result of deriving from its definition, as precise as possible and as general as necessary.

Every process component's technical name is derived from its name by

(a) removing blanks and
(b) using CamelCase.

Every process component definition must

(a) be understandable for a business user, not knowing SAP or partner context
(b) based on the abstraction of a real world topic
(c) be SAP agnostic, reflecting the proper industry related business meaning

3.4.3 Business Objects

3.4.3.1 Definition and Naming

In the following lists of business object aspects only the ones marked fat are subject to certification. All other aspects are required but will not be explicitly checked during certification. The vendor affirms that these requirements have been fulfilled to best knowledge.

A business object must be a specific view of well-defined and outlined business content which

(a) is designed free of redundant business functionality;

(b) represents a self-contained, independent **business concept, well-known and generally accepted in the business world;**

(c) is well determined in the context of other already available objects and the business processes it is involved in;

(d) describes a state before or after the execution of a process (step), and possible state transitions, but not the control of process (step) execution; and

(e) offers Service Operations which reflect the expected business needs to access or manipulate the object (or parts of it)

(f) is always assigned to the same process component.

Every business object's name must

(a) be commonly established or derived from the definition of the Business Object that is named;

(b) be singular

(c) be in British English

(d) have qualifiers always precede the name qualified;

(e) be in Title Case

(f) be, as a result of deriving from its definition, as precise as possible and as general as necessary.

If the business scenario requires country-specific specializations of business objects (e.g. in tax-related matters), the name of the specialized business object must be prefixed with the ISO country code in capital letters and a following space (e.g. "EN Payroll").

Every business object's technical name is derived from its name by

(a) removing blanks and

(b) using CamelCase.

Every business object definition must

(a) be stated in the singular;

(b) state what the concept is, not only what it is not;

(c) be stated as a descriptive phrase or sentence(s);

(d) contain only commonly understood abbreviations;

(e) be expressed without embedding definitions of other data or underlying concepts;

(f) state the essential meaning of the concept;

(g) be precise and unambiguous;

(h) be concise;
(i) be able to stand alone;
(j) be expressed without embedding rationale, functional usage, domain information, or procedural information;
(k) avoid circular reasoning;
(l) use the same terminology and consistent logical structure for related definitions;
(m) be not covered by definition of other Business Objects;
(n) not cover other Business Objects; and
(o) reflect the external view of the Business Object as a whole, and not just the internal structure.

(m) and (n) should be self-verified by the vendor against the existing SAP BOs in the Enterprise Services Workplace as well as own existing BO definitions.

3.4.3.2 Attributes and Visualization

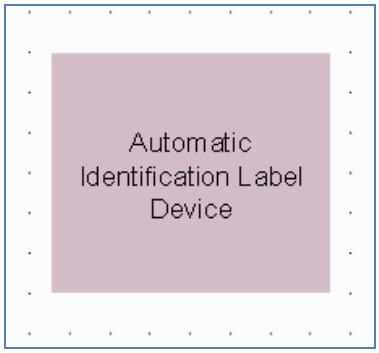
Each business object contained in the certification content is either to be of type
(a) BPO (Business Process Object),
(b) MDO (Master Data Object) or
(c) TO (Transformed object)

For every business object in the certification content the following model attributes must be maintained:

Model Attribute	**Value**
Name	Name
Description/Definition	Definition
SAP NetWeaver attributes – Business object type	Type

Visualization:

The BO name must be placed in the center and aligned centered within the service interface.



3.4.4 Service Interfaces

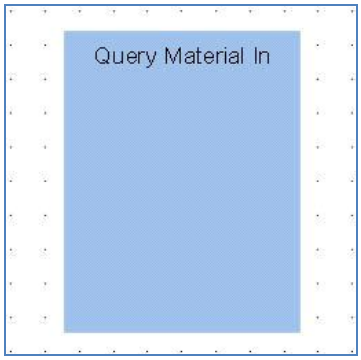
Every service interface must be an synchronous inbound A2X Service Interface (Outbound or A2A- and B2B-type Service Interfaces are not supported for certification).

Every A2X service interface name must follow one of the following patterns:

- Manage <BO name> In
- Query <BO name> In
- <BO name> Action In
- Entries for <BO name> In

Visualization: The service interface name must be placed at the top and aligned centered within the service interface.

Example:



Every service interface's technical name must be derived from its name by

(a) removing blanks,
(b) using CamelCase and
(c) adding the technical name of the containing process component technical name as prefix.

Example: Service interface name = "Query Material In" in process component "Product Data Maintenance" => Service interface technical name = "ProductDataMaintenanceQueryMaterialIn"

For each modeled service interface in the certification content the following attributes must be maintained:

Model Attribute	Value
Name	Name of the service interface according to the applicable interface pattern
Description/Definition	Description/Definition according to interface pattern.
SAP NetWeaver attributes – Technical name	Name of the service interface's technical name
SAP NetWeaver attributes – Direction	In=Inbound

3.4.5 Operations

3.4.5.1 General naming rules

Each operation must be assigned to exactly one business object and the appropriate service interface according to the corresponding interface pattern. The name of each operation must reflect the intended business purpose in accordance with the corresponding interface pattern.

The name of an operation (op) is usually based on the name of the operation-specific **service view**. The service view name must always contain the BO name.

Exception: The BO name can be omitted from the resulting operation name if the operation name is too long (tool restriction) since the BO name is always part of the interface name. In all other cases, the BO name must not be omitted (e.g. MT name and technical name).

3.4.5.2 "Manage" interface pattern: specific requirements

If the service interface name is of type *Manage <BO name> In*, every operation modeled within this service interface must be of one of the following types

Create, Update, Change, Cancel, Check, Read

Operation name and the SAP NetWeaver attributes "Message type request" and "Message type response" must adhere to the rules laid out in the respective tables below.

The vendor affirms that the operation implementation fulfills the corresponding business tasks accordingly.

Create	
Operation name	Create <service view name>
Business task	Creates a single instance of the business content described by the service view name, usually a single business object.
Message type request	<service view name> Create Request
Message type response	<service view name> Create Confirmation

Read	
Operation name	Read <service view name>
Business task	Reads a single instance of the business content described by the service view name, usually a single business object. The selection criterion uniquely identifies a single content instance and this instance is returned (allowed cases: "by ID" or, if a simple ID is not available: "by Identifying Elements")
Message type request	<service view name> by <Selection Criterion> Query [selection criterion: ID = identifier or "Identifying Elements"]
Message type response	<service view name> by <Selection Criterion> Response [selection criterion: ID = identifier or "Identifying Elements"]

Update	
Operation name	Update <service view name>
Business task	Requires a previous read-only access using the Read operation so that it can be checked within the Update operation if data has been changed in the meantime. The update can only be performed on unchanged data. The operation generally follows the rule for data-changing services (see 3.6.3 Requirements for Data-changing Services).
Message type request	<service view name> Update Request
Message type response	<service view name> Update Confirmation

Change	
Operation name	Change <service view name>
Business task	Overwrites data without checking whether any changes have been made to the data since the last read-only access. The Update operation checks this. The operation generally follows the rule for data-changing services (see 3.6.3 Requirements for Data-changing Services; see particularly section on Update vs. Change operations in A2X services).
Message type request	<service view name> Change Request
Message type response	<service view name> Change Confirmation

Cancel	
Operation name	Cancel <service view name>
Business task	Cancels a single instance of the business content described by the service view name, usually a single business object. This can involve a database deletion of the related data but usually involves setting of cancellation flags that trigger reversal processes as a consequence of the BO cancellation (e.g. the cancellation of a sales order might result in a complex set of actions in the logistics chain that has already been partly executed).
Message type request	<service view name> Cancel Request
Message type response	<service view name> Cancel Confirmation

Check	
Operation name	Check <service view name> [<suffix>]
Allowed suffixes	Creation, Update, Change, Cancellation
Business task	If no suffix is present, the data consistency of the sent data is checked. If a suffix is present the data consistency of the sent data is checked with respect to the data change operation denoted by the suffix (e.g. “Check Sales Order Creation” expects sales order data in the request. The response would simulate a Create Sales Order operation with this data and would return log information on the simulation success or failure). If a suffix is present the corresponding data change operation denoted by the suffix must exist in the service interface too.
Message type request	<service view name> [<suffix>] Check Query
Message type response	<service view name> [<suffix>] Check Response

The following table gives examples for all cases:

Examples:		
Operation name	**Request MT**	**Response MT**
Create Purchase Order	Purchase Order Create Request	Purchase Order Create Confirmation
Read Material	Material by ID Query	Material by ID Response
Update Production Planning Order	Production Planning Order Update Request	Production Planning Order Update Confirmation
Change Material	Material Change Request	Material Change Confirmation
Cancel Bank Card Contract	Bank Card Contract Cancel Request	Bank Card Contract Cancel Confirmation
Check Sales Order Creation	Sales Order Creation Check Query	Sales Order Creation Check Response

3.4.5.3 "Query" interface pattern: specific requirements

If the service interface name is of type *Query <BO name> In*, every name of an operation modeled within this service interface must start with the action *Find*.

Operation name and the SAP NetWeaver attributes "Message type request" and "Message type response" must adhere to the rules laid out in the table below.

The vendor affirms that the operation implementation fulfills the corresponding business task accordingly.

Find	
Operation name	Find <service view name> by <Selection Criterion> {and <Selection Criterion} [The operation name must not start with Find Allowed (see 3.4.5.5 "Entries for" interface pattern: specific requirements).]
Business task	Reads all instances of the business content described by the service view name that fit to the sent selection criteria, usually a set of single business object instances.
Message type request	<service view name> by <Selection Criterion> {and <Selection Criterion>} Query
Message type response	<service view name> by <Selection Criterion> {and <Selection Criterion>} Response

A selection criterion is a simple language construct but which can reflect a complex semantic situation, e.g.:

Operation name	**Business Functionality**
Find Employee Leave Request by Participant	Reads the latest versions of employee leave requests for a given approver or employee ID.
Find Physical Inventory Count by Material and Location	Reads the inventory counts of materials in locations (for both selection criterias value ranges must be specified).

Example: The operation should return purchase order items that have been ordered at a given date and which refer to products of a given product vendor.

=> service view = "Purchase Order Item"; Selection Criterion 1 = "Order Date"; Selection Criterion 2 = "Product Vendor" = "Vendor" ("Product" could be omitted because this is clear from the context).

=> Operation name = "Find Purchase Order Item by Order Date and Vendor"

3.4.5.4 "Action" interface pattern: specific requirements

If the service interface name is of type <*BO name*> *Action In*, every operation name and the SAP NetWeaver attributes "Message type request" and "Message type response" must adhere to the rules laid out in the table below.

The vendor affirms that the operation implementation fulfills the corresponding business task accordingly.

Action	
Operation name	<action> <service view name> <suffix> [The operation name must not start with one of the following actions: Create, Update, Change, Cancel, Read, Find]
Suffix	The action *Check* may only be used in conjunction with suffixes other than *Creation*, *Update*, *Change*, and *Cancellation*.
Business task	As semantically described by the operation name. In the case of a Check operation there must be a corresponding operation in the same Action interface with operation name <suffix> <service view name>. The Check operation checks if this corresponding can be successfully executed.
Message type request	<service view name> <action> <suffix> [Query|Request] [depending on communication pattern]
Message type response	<service view name> <action> <suffix> [Response|Confirmation] [depending on communication pattern]

Examples:		
BO name	**Operation name**	**MT names**
Utilities Invoice Request	Release Utilities Invoice	Utilities Invoice Request Release Request Utilities Invoice Request Release Confirmation
Document	Check In Document File Variant	Document File Variant Check In Request Document File Variant Check In Confirmation
Installation Point	Dismantle Individual Material	Installation Point Individual Material Dismantle Request Installation Point Individual Material Dismantle Confirmation
Material	Check Material Deactivation	Material Deactivate Check Query Material Deactivate Check Response

3.4.5.5 "Entries for" interface pattern: specific requirements

If the service interface name is of type *Entries for <BO name> In*, every name of an operation modeled within this service interface must start with the action *Find Allowed*.

Operation name and the SAP NetWeaver attributes "Message type request" and "Message type response" must adhere to the rules laid out in the table below.

The vendor affirms that the operation implementation fulfills the corresponding business task accordingly.

Find Allowed	
Operation name	Find Allowed <element name> by <Selection Criterion> {and <Selection Criterion>}
Business task	Is intended as value help functionality to return allowed values for the stated element. Reads all element instances (values) that are allowed given the sent selection criteria.
Message type request	<BO name> Allowed <element name> by <Selection Criterion> {and <Selection Criterion>} Query
Message type response	<BO name> Allowed <element name> by <Selection Criterion> {and <Selection Criterion>} Response

Example:		
BO name	**Operation name**	**MT names**
Employee Time Event	Find Allowed Cost Centre by Employee	Employee Time Event Allowed Cost Centre by Employee Query Employee Time Event Allowed Cost Centre by Employee Response

Please note: The term "Allowed" may as well be part of service view names in other interface patterns The difference lies in the intended functionality (Value help vs . business information).

3.4.5.6 Technical Names

Every service operation's technical name is derived from its name based on the following rule

Case 1:

If the implementation platform of the service interfaces described by the certification platform is SAP NetWeaver 7.0 ABAP (< SP14) [e.g. base of SAP ECC 6.0] or any other platform that has the restriction that per web service interface there can be only one operation, the technical name of an operation must be derived from the technical request MT name.

The *technical request MT name* is equal to the request MT name with blanks removed using CamelCase.

If the previously derived pair of request and response MT names end with Request/Confirmation:

Operation technical name =
<technical request MT Name>Confirmation_In

If the previously derived pair of request and response MT names end with Query/Response:

Operation technical name = <technical request MT name>Response_In

Example Query/Response pattern: Operation = “Read Material”

=> MTs “Material by ID Query” and “Material by ID Response”

=> Request MT name = “Material by ID Query”

=> Remove blanks and use Camel Case: technical request MT name = “MaterialByIDQuery”

=> Operation technical name = “MaterialByIDQueryResponse_In”

Example Request/Confirmation pattern: Operation = “Create Purchase Order”

=> MTs “Purchase Order Create Request” and “Purchase Order Create Confirmation”

=> Request MT name = “Purchase Order Create Request”

=> Remove blanks and use Camel Case: technical request MT name = “PurchaseOrderCreateRequest”

=> Operation technical name = “PurchaseOrderCreateRequestConfirmation_In”

Case 2:

In any other case the technical name of the operation must be derived from its name by

(a) removing blanks,
(b) using CamelCase and
(c) adding the technical name of the surrounding service interface as prefix with a separating dot [“.”].

Example: Above *Read Material* example in a non-SAP NetWeaver 7.0 ABAP implementation environment:

Operation technical name = ProductDataMaintenanceQueryMaterialIn.FindMaterialByID

3.4.5.7 Attributes and Visualization

For each modeled operation in the certification content the following attributes must be maintained:

Model Attribute	Content/Value selection
Name	Name of the operation
Description/Definition	Description/Definition of the functionality of the operation
SAP NetWeaver attributes – Technical name	technical name of the operation
SAP NetWeaver attributes – Message type request	Name of the message type of the request according to the applicable interface pattern
SAP NetWeaver attributes – Message type response	Name of the message type of the response according to the applicable interface pattern
SAP NetWeaver attributes – Mode	Synchronous

Visualization:

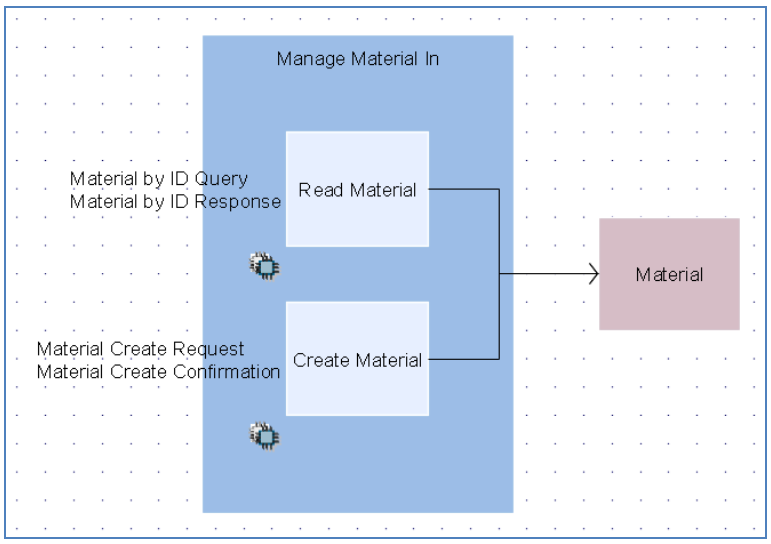
(1) The operation must be placed within the corresponding service interface. It must overlap with other service operations.

(2) The operation name must be located within the operation and placed in the center and aligned centered.

(3) Message type request and message type response must be placed above each other to the left of the operation with text aligned to the left (the corresponding properties of the operation must be maintained).

(4) Each operation **uses/ is realized** by the respective business object it belongs to (with connection occurrence).

Example:



3.5 Definition Content Requirements

Abbreviations	
Core Data Type	CDT
SAP Global Data Type	GDT
(Custom-defined) Business Data Type	BDT
Intermediate Data Type	IDT
Message Type	MT
Message Data Type	MDT

3.5.1 Service Interfaces, Operations and Model Assignments

Case 1:

If the implementation platform of the service interfaces described by the certification platform is SAP NetWeaver 7.0 ABAP [e.g. SAP ECC 6.0] or any other platform that has the restriction that per web service interface there can only be one operation:

For each modeled operation there must exist one service interface definition in a valid vendor namespace with its name being equal to the operation's technical name.

For each modeled operation there must be an operation assignment to the operation in the corresponding service interface definition. There must be no service interface assignment.

For each operation definition the *Description* field must be maintained. It must contain the same content as the description/definition attribute of the corresponding modeled operation.

Case 2:

In any other case:

For each modeled service interface there must exist one service interface definition in a valid vendor namespace with its name being equal to the modeled service interface's technical name.

For each modeled operation there must be an operation in the service interface definition that corresponds to the modeled service interface containing said operation and its name must be equal to the modelled operation's technical name without the preceding prefix of the service interface technical name.

For each modeled service interface there must be a service interface assignment to the corresponding service interface definition.

For each modeled operation there must be an operation assignment to the corresponding operation in the containing service interface definition.

For each service interface definition the *Description* field must be maintained. It must contain the same content as the description/definition attribute of the corresponding modeled service interface.

For each operation definition the *Description* field must be maintained. It must contain the same content as the description/definition attribute of the corresponding modeled operation.

In the current version of the ESR content certification, the following service interface definition attribute value restrictions apply:

Definition Attribute	Allowed Values
Category	Inbound
Interface Pattern	Stateless Stateless (XI30-Compatible)

In the current version of the ESR content certification, the following operation attribute value restrictions apply:

Definition Attribute	Allowed Values
Operation Pattern	Normal Operation
Mode	Synchronous

3.5.2 Naming of Request, Response, Fault Message Types and related Message Data Type

Request messages must be typed by a **Message Type** (MT) with a name equal to the request MT name of the corresponding modeled operation with

(a) blanks removed and
(b) applied CamelCase and
(c) concatenated suffix _sync

Example: Request MT name in model = Sales Order by ID Query => MT name in definition = SalesOrderByIDQuery_sync

Response messages must be typed by a MT with a name equal to the response MT name of the corresponding modeled operation with

(a) blanks removed and
(b) applied CamelCase and
(c) concatenated suffix _sync

Example: Response MT name in model = Sales Order Create Confirmation => MT name in definition = SalesOrderCreateConfirmation_sync

For each request and response MT definition the property *Description* must be maintained. For certification, English descriptions must be available.

Each request and response MT definition must use a corresponding **Message Data Type** (MDT) derived from the MT name by inserting the qualifier “Message” before the suffix _sync.

Example: MT definition name = SalesOrderCreateConfirmation_sync => MDT name = SalesOrderCreateConfirmation**Message**_sync

Fault messages must be typed by the SAP-standard fault MT *StandardMessageFault* using the SAP-standard fault MDT *ExchangeFaultData*.

3.5.3 Top-level Message Data Type Structure

Each MDT must contain the following direct sub-nodes in the stated order depending on the respective communication pattern of the containing operation. The business information is held in the *content root node* which acts as a container for all exchanged business data.

Query-Response Communication Pattern

Note: The node ProcessingConditions generally used for navigating in result sets is generally optional but if included must be placed according to the below pattern (in both the query and the response message). The GDTs NOSC_QueryProcessingConditions and NOSC_ResponseProcessingConditions may be replaced by own business data types if functionally necessary. If so, the data type names must have a prefix "PARTNER_" and must end with " ProcessingConditions". The node name "ProcessingConditions" remains the same.

Query message data type			
Parts	**Name**	**Occurrence**	**Data Type**
Content root node	[see next chapter]	1	Message specific IDT (naming see below)
Processing Conditions	ProcessingConditions	0..1	NOSC_QueryProcessingConditions

Response message data type			
Parts	**Name**	**Occurrence**	**Data Type**
Content root node	[see next chapter]	0..unbound (Find) 0..1 (Check/Read)	Message specific IDT (naming see below)
Processing Conditions	ProcessingConditions	0..1	NOSC_ResponseProcessingConditions
Log message	Log	1	NOSC_Log

Request-Confirmation Communication Pattern

Request message data type			
Subnode	**Name**	**Occurrence**	**Data Type**
Message header	MessageHeader	0..1	NOSC_BasicBusinessDocumentMessageHeader
Content root node	[see next chapter]	1	Message specific IDT (naming see below)

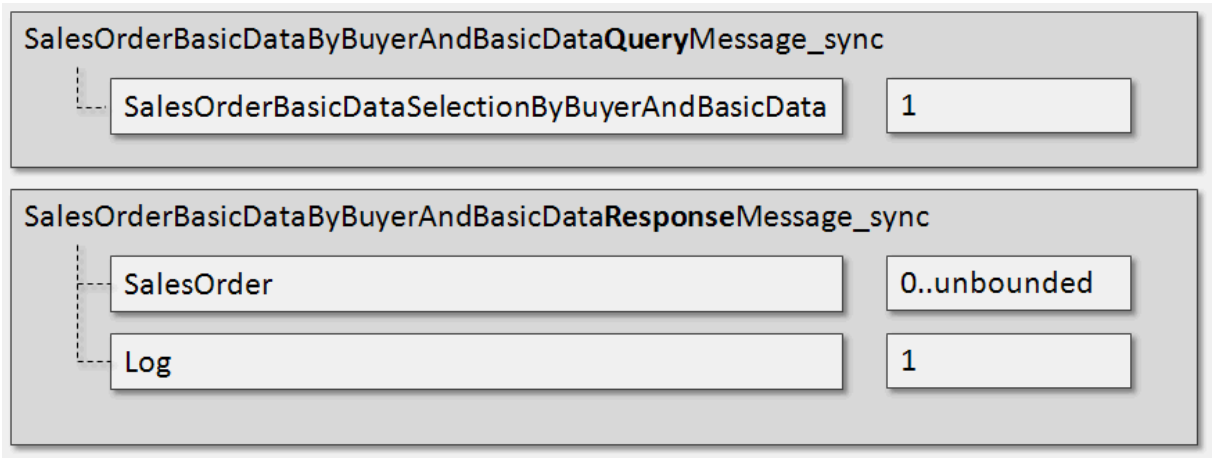
Confirmation message data type			
Subnode	**Name**	**Occurrence**	**Data Type**
Message header	MessageHeader	0..1	NOSC_BasicBusinessDocumentMessageHeader
Content root node	[see next chapter]	0..1	Message specific IDT (naming see below)

Log message	Log	1	NOSC_Log

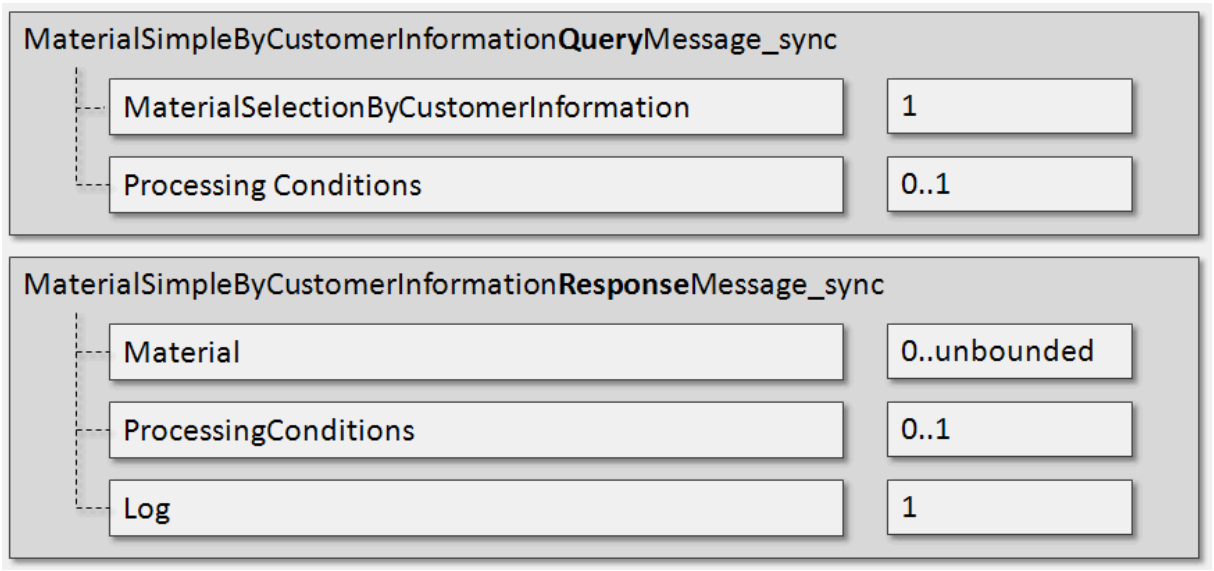
If used, the predefined data types NOSC_BasicBusinessDocumentMessageHeader, NOSC_Log, NOSC_QueryProcessingConditions, and NOSC_ResponseProcessingConditions must be copied to the local namespace from the appropriate GDT software component version delivered by SAP (currently SAPGLOBAL 2.0). At time of compilation of the document at hand they can be found in the namespace http://sap.com/xi/SAPGlobal/GDT in the subfolder “SAP Business Suite additional”.

Examples:

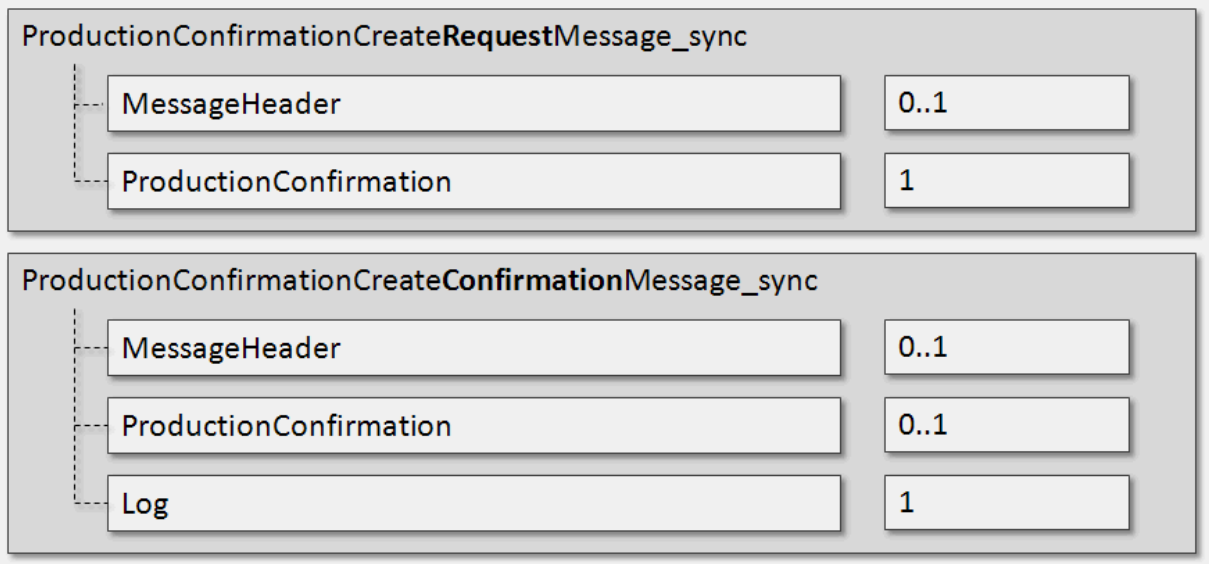
Query-Response Communication Pattern without Processing Conditions



Query-Response Communication Pattern with Processing Conditions



Request-Confirmation Communication Pattern



3.5.4 Content Nodes and Elements

3.5.4.1 Content root node naming

The naming of the content root node of a Message Data Type (MDT) depends largely on the associated communication pattern. The following cases must be distinguished:

- Request/Confirmation/Response messages and Check operation messages
- Query messages for other than Check operations

Although Check operations follow the Query/Response communication pattern they are treated like an operation in the Request/Confirmation pattern.

The is because a Check operation simply "simulates" an associated manage operation (Create, Change, …). So the content root node structure would be the same as for the associated manage operation. The operation returns all log messages and returns maybe default values for fields in the sent BO structure but makes no updates to the database.

3.5.4.1.1 Request/Confirmation/Response messages and Check operation messages

The content root node name must be equal to the business object technical name.

Example: Sales Order Item by Product and Delivery Date Response =>
Content root node name = "SalesOrder"

Special case: In operations following the "Entries for" pattern the content root node name is equal to

<BO name>Allowed<element name>

as it is used in the technical MT name.

Example: Employee Leave Request Allowed Approver By Identifying Elements Response =>
Content root node name = " EmployeeLeaveRequestAllowedApprover"

3.5.4.1.2 Query messages for other than Check operations

In a query MDT selection criteria are defined within a **selection view**. In this case the MT name for the response message has the following general structure:

<view part> by <selection criteria part> Query

Example: Sales Order Item by Product and Delivery Date Query =>

<view part> = “Sales Order Item”; < selection criteria part > = “Product and Delivery Date”

The content root node name must be derived from the MT

<view part> Selection by <selection criteria part>

by

(a) removing blanks and
(b) applying CamelCase

Example: Message type = “Sales Order Item by Product and Delivery Date Query”

=> Content root node name = “SalesOrderItemSelectionByProductAndDeliveryDate”

3.5.4.2 Message element naming and typing

Every element of a message data type must be typed. All elements must be typed by either by a CDT, GDT or a custom-defined BDT.

Nodes that are introduced for structuring the message data type (including the content root node) must be typed by IDTs.

Elements must not be directly typed by simple XSD types!

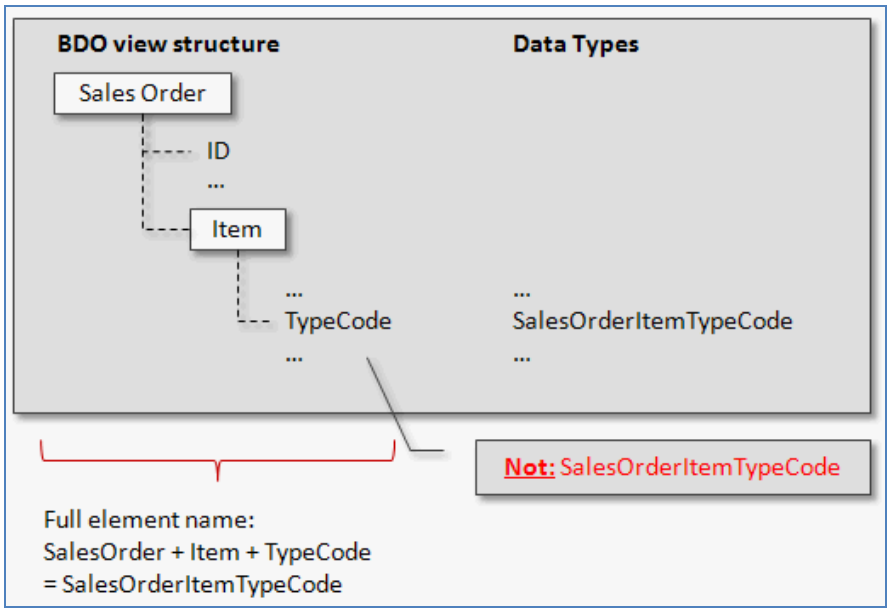
3.5.4.2.1 Naming message elements typed by CDT, GDT or BDT

If the message element is typed by a CDT, GDT or BDT the element’s name must be derived from the data type name according to the following rules:

Element Name = <Additional Qualifiers> + <DT name>

Names of nodes containing an element must be omitted from the resulting name from the beginning to avoid redundancy.

Example:



Special case:

If a leaf element is typed by a structurally restricted data type (for details refer to the Interface document), the element name must be derived according to the following rule:

Restricted DT name = <VARIABLE>_<DT name>

=> Element Name = <Additional Qualifiers> + <DT name>

In other words: The variable name specifying the restriction is omitted.

Example: Status + MEDIUM_Text = StatusText

3.5.5 Data Types

3.5.5.1 Naming rules for Business Data Types

If no suitable GDT is available for typing an element, either an own BDT must be created, or a (simple) CDT must be used. A simple BDT (i.e. a BDT without a sub-structure) must be derived from a CDT i.e. it has the *Classification* "Core Data Type" and the *Representation Term* corresponding to the intended CDT.

Please note that in A2X service interface definitions supplementary components inherited from CDTs must be removed (for details see 3.5.5.2 NOSC Data Types).

The name has to consist of qualifiers in British English that are concatenated by Camel Case.

Soft rule/Guideline: Each qualifier must be a commonly understandable term in the addressed business domain. The BDT must have a message-independent meaning which allows it to be re-used across several messages/operations.

In a way such a BDT is a data structure template for a business meaning. Message-specific aggregated DTs are called Intermediate Data Types (page 32).

The name of a BDT must contain at least two terms/qualifiers: [Object class] followed by its [Representation]. The object class can further be specified by means of a [Property] term, positioned in-between the object class and the representation. Object class, property and representation must be concatenated using CamelCase. Each term may have a preceding Qualifier. Redundancies within the name as well as to the root node must be omitted.

Example: Object class=shirt, Property=size; Representation=Code
=> BDT ShirtSizeCode

The object class name is the name of the object for which the data type is used for typing the object's properties. The property refers to a specific attribute of the object class that needs to be specified/ described. The representation name is the name of the CDT that the simple BDT has been derived from.

Both [Object class name] and [Property name] can themselves be further specified by qualifiers to reflect the full semantic.

Example: Object class=business partner, Property=type; Representation=Code
=> BDT BusinessPartnerTypeCode

If the BDT name resulting from above rule shows redundancies, the redundancies must be removed. This will mostly occur if [Property name] already contains the [Representation name] as ending qualifier.

Example: Object class=person, Qualifier = first Property= name; Representation=Name
=> BDT PersonFirstNameName => remove redundancy => BDT PersonFirstName

A simple BDT which refers to the object class of a business object must use the business object name as object class name

Example: Business object = Object class= business transaction document, Property=type; Representation=code
=> BDT BusinessTransactionDocumentTypeCode

A simple BDT which describes a property of an object class of a business object node must use the containing business object name concatenated with all names of the business object node hierarchy using CamelCase as object class name.

Example:

1. Business object = Sales Order, business object node = Item
   Object class=sales order item, Property=status; Representation=code
   => BDT SalesOrderItemStatusCode

2. Business object node hierarchy = Sales Order -> Item -> Delivery Schedule
   Object class=sales order item delivery schedule, Property=type; Representation=code
   => BDT SalesOrderItemDeliveryScheduleTypeCode

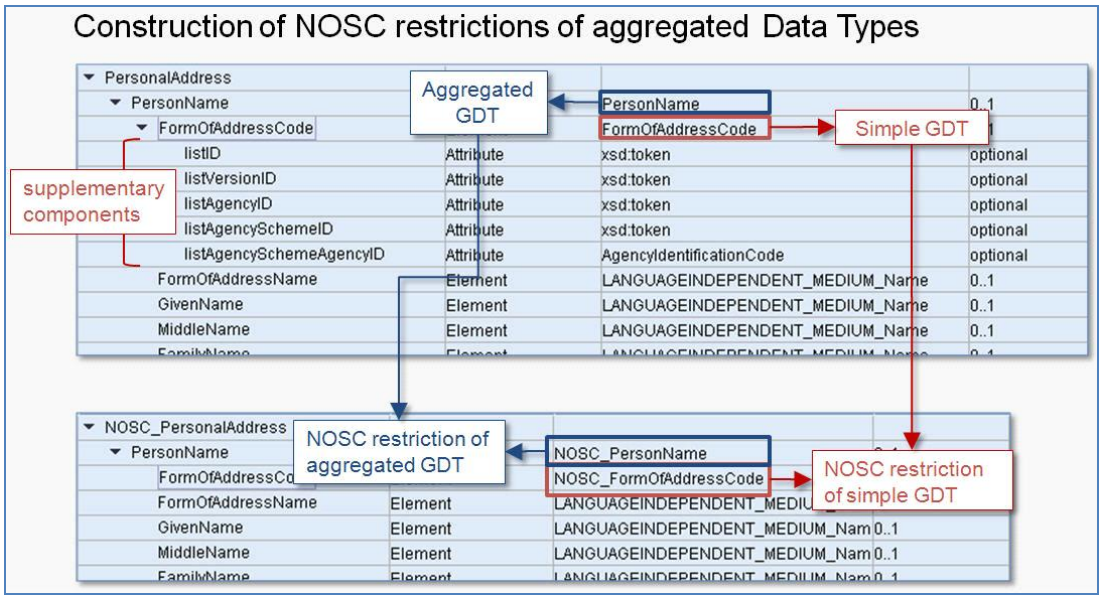
3.5.5.2 NOSC Data Types

Data types that are required for typing elements in MDTs of A2X service interfaces must not carry any supplementary components, neither themselves nor any of their sub-elements. If such a data type is nevertheless semantically appropriate, an NOSC (No supplementary components) restriction of this data type must be created and used instead.

If the data type is a simple data type a data type copy must be created in the vendor's Software Component Version and namespace that is equal to the original with all supplementary components removed. The name of this data type must be equal to the original name with the prefix NOSC_ (no supplementary components) added.

If the data type is an aggregated data type that contains leaf elements carrying supplementary components this procedure must be applied iteratively from the root down to all data types of such leaf elements.

Example:

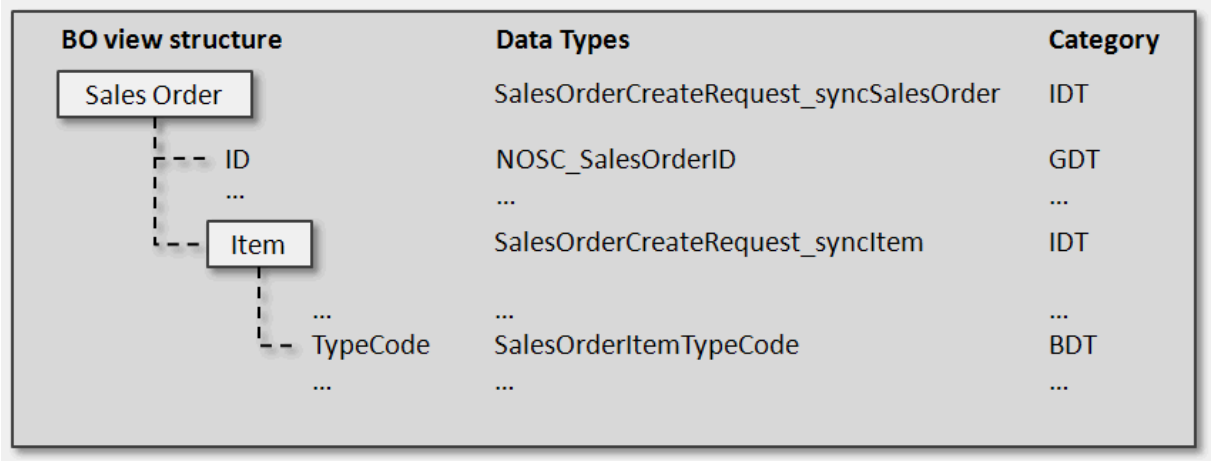


3.5.5.3 Intermediate Data Types

Nodes that are introduced for structuring a content node must be typed by IDTs. The type name is derived based on the following rule:

[MT name] + [node name unique within the MDT]

Example: MT SalesOrderCreateRequest_sync, sub-node Item
=> Type name for Item: SalesOrderCreateRequest_syncItem



As these names can become very long they may be abbreviated according to the abbreviation rules.

3.5.6 Miscellanious

3.5.6.1 Response messages: Code-type elements with corresponding natural language text

For each simple element that is returned in a response or con MDT and that is typed by a data type derived from the CDT Code there must be a simple element typed by a data type derived from the CDT Name (or restrictions of it).

The vendor affirms that such name elements return the name of the associated code value requirement.

The name of this element must equal to the name of the code type element with the suffix *Code* replaced by *Name* and the element must directly follow its associated code type element in the response message data type.

Example:

Returned code-like element: DivisionCode typed by GDT Divison Code

=> Returned name-like type element: DivisionName typed by MEDIUM_Name

Material	Element	MaterialBasicDataByIDResponse_syncMatl	0..1
InternalID	Element	NOSC_ProductInternalID	1
TypeCode	Element	NOSC_MaterialTypeCode	1
TypeName	Element	MEDIUM_Name	1
IndustrialSectorCode	Element	NOSC_IndustrialSectorCode	1
IndustrialSectorName	Element	MEDIUM_Name	1
PreviousName	Element	LANGUAGEINDEPENDENT_MEDIUM_Name	0..1
DeletedIndicator	Element	Indicator	0..1
DivisionCode	Element	NOSC_DivisionCode	0..1
DivisionName	Element	MEDIUM_Name	0..1

Recommendation: It is a good practice to do the same for IDs and to send descriptive text for the respective ID if available and applicable for the intended use case.

3.6 Implementation Requirements

The vendor affirms that the following implementation requirements have been fulfilled to best knowledge.

3.6.1 Transactional Behavior (Statelessness, Atomicity)

Each service interface must be implemented such that each call to one of the operations represents an atomic transaction from the provider's perspective, and such that no consumer application session context is held in-between service calls.

3.6.2 Reliable Execution (QoS "Exactly Once", Consistency)

Synchronous services to create or modify data should provide special measures to prevent the exact same request being executed multiple times – they should be implemented as idempotent receivers:

- Their request message contains a MessageHeader element typed with SAP GDT NOSC_BasicBusinessDocumentMessageHeader (or a restriction thereof that contains the UUID element).
- Once a request message has been processed successfully, the service responds to the same request message (as identified by the MessageHeader/UUID element's value) received again within a limited timeframe with the original response, without executing the business logic again.
- If the service is modeled as idempotent, but the behavior described in the aforementioned rule cannot be implemented, the service returns a fault message if it receives a request message with a MessageHeader/UUID value.

If an operation implementation contains database updates, a response to a operation call must consistent with the database state. In other words, such that the service consumer immediately querying the data again after it received the response finally has two consistent responses (assuming the data is not modified between the subsequent service calls through other channels).

The GDT NOSC_BasicBusinessDocumentMessageHeader is documented in detail in the Enterprise Services Workplace at http://esoadocu.sap.com/socoview(bD1lbiZjPTgwMCZkPW1pbg==)/render.asp?packageid=DBBB6D8AA3B382F191E0000F20F64781&id=69DA011F3B6611DC5A11000F20FCB6A9 .

3.6.3 Requirements for Data-changing Services

There are generally two different strategies for data-changing operations:

For the overwrite strategy, also called *last-one-wins strategy*, each database modification attempt is accepted and the data provided simply overwrites the old data in the database.

For the optimistic locking strategy, also called *first-one-wins strategy*, only the first one of a sequence of consecutive database update attempts will be successful, so the data provided in the first attempt will persist in the database, whereas further attempts will fail.

Taking the example of a updating a stock amount, according to the *first-one-wins strategy*, every database update attempt following the first one fails, and the application doing this consecutive service call is forced to query the current stock again and re-calculate the new total stock amount.

A data-changing operation must adhere to the following correlated naming and implementation convention:

- If the operation name contains “Update” as action name, the service and the corresponding read, create and cancel services cooperate to detect concurrency conflicts using optimistic locking (first-one-wins strategy).
- If the operation name contains “Change” as action name, the implementation does not detect concurrency conflicts, but always overwrites (last-one-wins strategy).

“Update”-operations should be preferred for A2X services. “Change” services should be only provided for use cases where the data is clearly owned by the service consumer and the “last-one-wins” strategy is acceptable from the targeted business processes’ perspective. Especially for Master Data Business Objects change services should never be provided.

A data-changing operation must follow the following additional rules:

1. The absence of an optional, single-valued element (maxOccurs=1) in an XML message does not have any particular meaning – in particular it is not interpreted as a request to reset the corresponding field in the service provider’s database.
2. Two ways of modifying the values of multi-valued elements (maxOccurs>1) are supported if the corresponding line items contain a stable identifier:

   By default, only the line items contained in a message received are modified. The line item structure contains an actionCode attribute (typed with SAP GDT ActionCode) to control whether the identified line item is to be added, changed or cancelled.

   Passing value “true” for a <name of line item element>ListCompleteTransmissionIndicator attribute (typed with SAP GDT Indicator) of the line item’s parent element, consumers can indicate that the entire set of line items to remain, with the modified values, is passed. In this case, line items on the database, but not in the message received, are to be cancelled.

If this second rule cannot be followed, e.g. because the line items do not contain a stable identifier, the service’s behavior is documented. It is classified as implementing one of the change/update behavior types as described in SAP Note 1007799 and the SAP Enterprise Services documentation (see Change/update behavior type 1 and Change/update behavior type 2).

3.6.4 Exception and Error Handling (NOSC_Log and StandardFaultMessage)

SAP distinguishes between two classes of error situations a service implementation can encounter during runtime, and that have to be treated differently to qualify a service as Enterprise Service:

- Technical failures, often referred to as exceptions
- Business-level errors and conflicts, often simply referred to as errors.

The implementation to be certified should follow this distinction as described in detail below. If the implementation requires including calls to backend functionality that does

not make this distinction, which in turn might lead to business-level messages that underneath originate in a technical failure in the backend, the certification does not require that the implementation is resolves this issue although it is recommended.

1. Technical Failures

Technical Failures, that is Exceptions, signal abnormal situations not expected by the service implementation and that a typical business user of the service consumer or provider application cannot solve. They usually require a system administrator, or software developer or customer support to be resolved. Examples:

- Bugs in provider or consumer application code (service contract violations)
- System errors / exceptions due to unavailable resources etc.

The unexpected situations occur very rarely in productive environments, provided consumer and provider and their configuration have been tested reasonably well.

Each Enterprise Services must indicate such failures using a Fault Message. The fault message must be named StandardMessageFault and have the same structure as the corresponding Fault Message Type in SAP namespaces containing Enterprise Services.

2. Errors and conflicts

Errors and conflicts indicate violations of business rules and are caused, for example, by wrong values in technically valid input messages, missing provider configuration, or temporary issues, for example concurrency conflicts. Such issues are frequent in day-to-day business. They should thus be treated as expected, though “unhappy” situations. They can usually be resolved by typical business users.

Synchronous Enterprise Services shall communicate such errors and conflicts back to the service consumer in the normal response message, using an element named Log and with the same structure as the SAP Global Data Type Log (NOSC_Log without supplementary components for A2X Services). You may find the detailed documentation for this GDT in the ES Workplace (http://esoadocu.sap.com/socoview(bD1lbiZjPTgwMCZkPW1pbg==)/render.asp?packageid=DBBB6D8AA3B382F191E0000F20F64781&id=82553381579F11DB626E001125F52824).

The Log’s sub-element BusinessDocumentProcessingResultCode shall convey whether a service request was processed successfully or not, using the valid values as defined by SAP.

Remark/Recommendation: Asynchronous Enterprise Services should trigger an error and conflict resolution workflow in the service provider application, if possible.

References

[1] Enterprise Services Repository for SAP NetWeaver CE.

[2] Configuring, Working with and Administering System Landscape Directory.

[3] Content Model Types.

[4] Transferring Design Objects.