Enterprise Services for SAP Customer Relationship Management

Release 5.1 SP03

Copyright

Icons in Body Text

Icon	Meaning
⚠	Caution
❖	Example
💡	Note
⬆	Recommendation
⟨⟩	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help* → *General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Typographic Conventions

Type Style	Description
Example text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
`Example text`	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Enterprise Services for SAP Customer Relationship Management

SAP Enterprise Services

Purpose

Enterprise Services from SAP afford you many new opportunities to automate your business processes and react flexibly to changes in your business processes. You can convert new business processes much more easily. You can also implement the *Enterprise Services* in heterogeneous, distributed system landscapes by implementing standardized Web technologies.

Integration

- A large number of *Enterprise Services* for typical business processes are already available in the *SAP Business Suite* applications. You can implement these services so that they communicate directly with other applications, client programs, user interfaces, and so on.
- Alternatively, you can use the *Enterprises Services* from SAP using the *SAP NetWeaver Exchange Infrastructure* (*SAP NetWeaver XI*). In this case, *Enterprise Services* communication occurs using the *Integration Broker* of the *SAP NetWeaver Exchange Infrastructure*, which contains additional functions for the processing and creation of messages. In the *Enterprise Services Repository* in *SAP NetWeaver XI* you can find descriptions of the interfaces of all the services that SAP has developed for communication with SAP applications. You can also define your own *Enterprise Services* using the *SAP NetWeaver Exchange Infrastructure*.

Features

- *Enterprise Services* are standardized interfaces for specific functions within an application, for example, for functions that read the business partner's address or that update reports. They simultaneously display descriptions of process steps and technical modules. In this way, and due to their very specific syntax and strictly defined naming convention, the *Enterprise Services* represent a shared language for communication between different business areas (for example, marketing and IT).
- You can use the *Enterprise Services* from SAP to create user interfaces independently of the underlying individual applications in the back-end system. These interfaces can be exactly adjusted to the workflows and work organization of your company.

  You can therefore provide user interfaces that are tailored exactly to the tasks and knowledge of the employees in your company. You use the same services and process the same data regardless of whether you create user interfaces for experts or for occasional users. The *Enterprise Service* works in the background on the technical details and data entry and processing is considerably simplified.
- With the help of the *Enterprise Services* from SAP, you can trigger automated processes, enter and process data, and exchange data with other systems and applications. In contrast to simple Web services, *Enterprise Services* from SAP are generally logical steps of a business process and can thus be used to support cross-application and cross-departmental business processes.
- You can implement *Enterprise Services* from SAP to automate flows for which you previously could not use any standard SAP solution or for which there previously was

no integration possible. SAP does not only develop *Enterprise Services* with the aim of providing individual interfaces for communication between individual application functions. *Enterprise Services* from SAP also aim to enable central business processes using several *SAP Business Suite* applications. For this purpose, *Enterprise Services* are bundled into *Enterprise Services Bundles* (ES Bundles). The *ES Bundles* represent templates for the processing of *Enterprise Services* for the illustration of business processes such as Order to Cash. You can find descriptions of the *Enterprise Services Bundles* in the *Enterprise Services Workplace* (ES Workplace) by choosing **`sdn.sap.com`** → *Enterprise SOA* → *ES Workplace* → *ES Bundles Wiki.*

- The *Enterprise Services* are used to split the complex functions of a business process into reusable elements. These elements – the individual *Enterprise Services* – can then be reused, exchanged, combined into new combinations, or incorporated into completely different business processes according to your own needs.

  Let us assume that you use the *Enterprise Services* to provide both an application with which your employees can submit their expenses and a number of additional services for the distribution of expenses. At the moment, the data is probably sent to particular administrators in the relative departments. However, you already anticipate that a central administration center will take over the processing in the near future. An automatic system for processing expenses or outsourcing to a foreign company is probably under discussion.

  If you use *Enterprise Services* you need only perform such changes in a few critical places. In addition, nothing will change on the employee's user interface, regardless of how you construct the distribution in the background. The function for sending expenses remains unchanged for employees and they are thus not required to understand the various flows in the background.

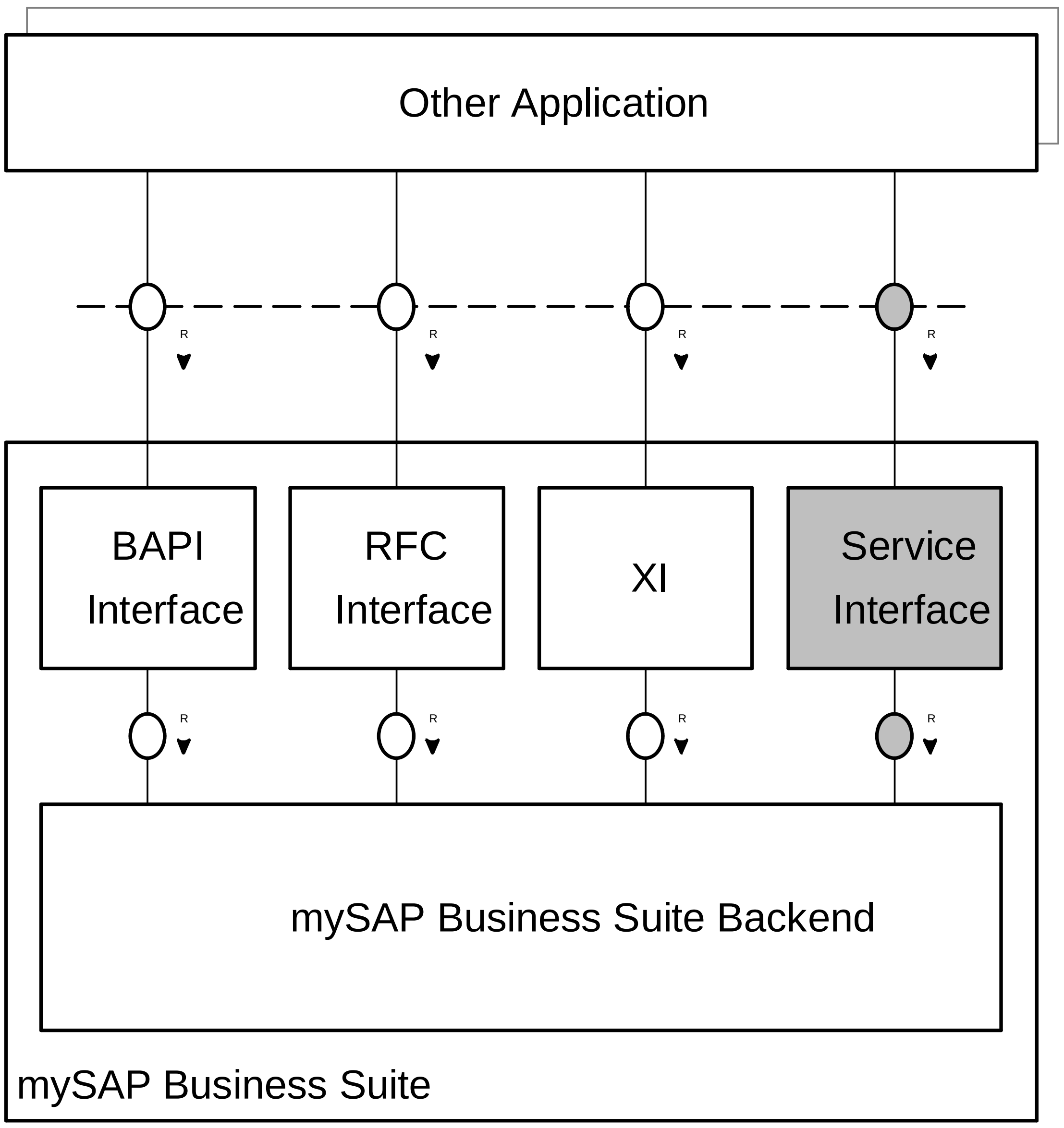
mySAP Business Suite: Service Processing

With the product enterprise services packages for the mySAP Business Suite 2005, SAP is moving the mySAP Business Suite 2005 toward Enterprise Service-Oriented Architecture (SOA). Introducing Enterprise SOA into the mySAP Business Suite 2005 naturally has implications for its software architecture. This document describes the implications and the concepts used to meet the challenges that arise.

Introduction to Enterprise Service-Oriented Architecture

The introduction of Enterprise Service-Oriented Architecture (SOA) essentially constitutes a new, more standardized way of accessing mySAP Business Suite functionality from other applications, alongside conventional access mechanisms like Business Application Programming Interfaces (BAPIs), Remote Function Calls (RFCs) and the SAP Exchange Infrastructure (SAP XI). Another application in this context may be an independent third-party application, a composite application or another SAP application like another mySAP Business Suite instance.

Introduction of Enterprise SOA into the mySAP Business Suite



Implementation Considerations

Enterprise SOA sets requirements for functionality access both on a technical and semantic level. To meet the requirements, which are detailed in the following sections, SAP has built a service interface layer on top of the mySAP Business Suite as shown in the figure above.

When discussing the SOA-enabling of the mySAP Business Suite, three different dimensions need to be distinguished and addressed:

- *Runtime:* the running system including all mechanisms in use to process a request via the service interface layer

  See mySAP Service Provisioning: Runtime

- *Design time:* procedures and methods applied by SAP during application design and development to create the software in the first place

  See mySAP Service Provisioning: Design Time

- *Configuration:* all concepts concerning activation and publication of functionality provided by the service interface layer

See mySAP Service Provisioning: Configuration

Note

It is expected that the reader is familiar with the structure and the basic architecture concepts of the mySAP Business Suite. For detailed information on the mySAP Business Suitesee the SAP help portal (help.sap.com *mySAP Business Suite*) or the SAP Developer Network (SDN) (sdn.sap.com).

End of the note.

Features

The remainder of this section defines the relevant terms and introduces the concept of enterprise SOA:

- Service
- Web Service
- Enterprise Service
- Service-Oriented Architecture
- SAP's Enterprise SOA

Example

The section Example: Service Operation ‘Create Employee Leave Request’ provides a walk-through of service provision.

Composite Application

A composite application is an application that is built upon the mySAP Business Suite and uses and extends the functionality of the mySAP Business Suite. Composite applications are developed by both SAP and external companies.

Service

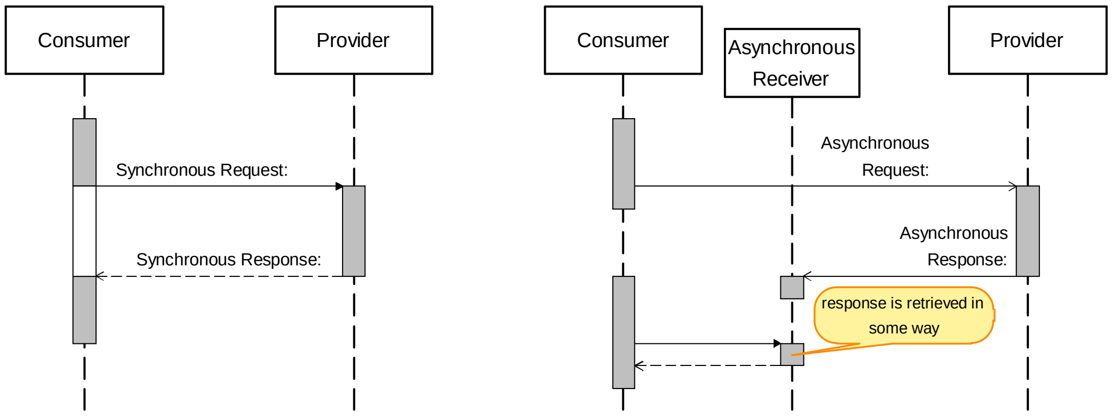
A service is a well-defined piece of functionality which can be invoked via a stable and implementation-independent interface.

The entity that offers a service to other entities is called *service provider*. An entity that makes use of a service by triggering its execution and – if appropriate – receiving its result, is called *service consumer*.

Synchronous/Asynchronous Communication

Services may be invoked either *synchronously* or *asynchronously*. In the synchronous case, the service consumer waits for the result after triggering the service provider, and is blocked until the result has arrived. In the asynchronous case, the service consumer continues its work after triggering the service provider, and may go back to the original task, as soon as the answer arrives. For the asynchronous case, an additional agent for receiving the response and a well-defined interface for the return of the response are required.

Synchronous Versus Asynchronous Service Call



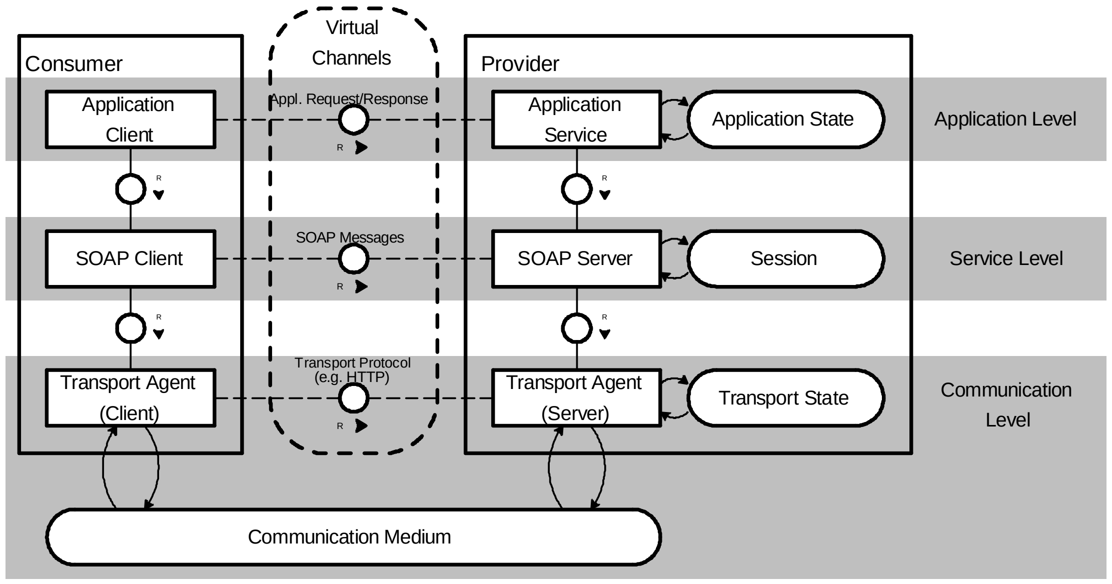
Statefulness/Stateless

Services may be *stateful* or *stateless*. Statefulness means that context information is stored on the provider side between successive service calls. State may be established on different levels as shown in the figure below.

- On the *communication level*, by means of a stateful protocol.
- On the *service level*, by combining successive service calls from the same consumer to a session (for example by exchanging session IDs).
- On the *application level*, by using application-specific mechanisms.

A stateless service, as defined by SAP, is a service that is stateless on the service level. However, it may establish state on the communication level or application level.

Levels of Statefulness



Web Service

A Web service is a special service which can be triggered via a network and conforms to the following Web standards:

- *Web Service Definition Language (WSDL)* is a standardized XML format for describing a service. In WSDL, the term service refers to a concrete instance of a service, provided by a service provider at a specified network endpoint. A service description in WSDL therefore comprises a service interface description and a concrete binding.

  The service interface is the abstract technology-independent description of the service. In WSDL, a service may comprise several service operations which can be invoked separately. For each operation WSDL describes the parameters needed to invoke the service and the result that will be returned.

  The binding determines the technical parameters, such as the network endpoint that acts as the service provider, the protocol used to communicate with the service provider and the format in which messages are exchanged.

- *Universal Description, Discovery and Integration (UDDI)* is the name of a directory service which can be used to publish the WSDL descriptions of the service instances that a service provider wants to make available to other applications.

- *SOAP* is a standardized, message-based protocol that is used by Web services for communication between the service consumer and the service provider. A SOAP message comprises a header part, which contains communication control parameters, and a body part, which contains the payload

The requirement of adherence to these three standards makes a Web service a standardized form of an Internet-based service.

Service-Oriented Architecture

Service-Oriented Architecture (SOA) is the term for software architecture that is based on the concept of services.

Structure

In SOA there are one or more *service providers* on one side, and one or more *service consumers* on the other side.

The service providers offer certain functionality as services and publish their interfaces which are designed to be stable.

Service consumers and service providers are only loosely coupled. The interplay of consumer and provider is based on the service interface definition as a kind of contract. The concrete implementation of a service is hidden from the consumer.

The services should be defined in such a way that they can be found and used easily in various applications. Consumers usually make use of several services, which may come from multiple service providers to create new functionality.

Benefits

Following the principles of SOA when building an application provides several advantages:

- *Flexibility*

Since service interfaces according to SOA are stable, and service implementations are transparent for the consumers, SOA provides the prerequisites for applications in which higher-level functionality realized by the service consumer can be modified without having to change the underlying service functionality.

- *Productivity*

  If the services are well defined, it is very efficient to build and advance SOA-based applications once a pool of services is available.

- *Adaptability*

  SOA facilitates the integration of different applications, thanks to the use of open and non-proprietary standards. This helps customers to integrate applications from different application vendors into their IT landscape.

Enterprise Service

An enterprise service is a service which provides business functionality and which is published by SAP in the enterprise services workplace in the SAP Developer Network (SDN) (at ▸sdn.sap.com ▸*Enterprise SOA* ▸*ES Workplace* ◂). In general, enterprise services have the following features:

- *Business semantics:* Enterprise services are structured according to a harmonized enterprise model based on process components, business objects and global data types (GDTs). They are defined using an outside-in approach: common business rules and know-how, rather than SAP-specific implementations, are the guideline for defining the business content of SAP applications.

- *Quality and stability:* Enterprise services ensure a stable interface for future versions (backward compatibility). Their behavior, prerequisites, dependencies of usage and configuration possibilities are well documented.

- *Standards:* Enterprise services are based on open standards. The interfaces are described according to WSDL. They are created out of global data types which are based on UN/CEFACT CCTS (Core Component Technical Specification).

There are different application areas for enterprise services, for example application-to-application (A2A) and business-to-business (B2B) integration or user interfaces.

Integration

Enterprise Services for the mySAP Business Suite 2005

A first wave of enterprise services for the mySAP Business Suite was released in the product enterprise services packages for the mySAP Business Suite.

While enterprise services can have different technical characteristics, most of the enterprise services in the enterprise services packages are optimized for a usage in which SAP makes no assumptions about the service consumer. As a consequence, these enterprise services are as simple and easy to use as possible, which results in the following characteristics:

- *Atomic transaction:* The execution of a service must always result in consistent business data. It shall not be necessary to execute any further services subsequently to achieve a consistent state on the database.

- *Statelessness:* No context information is stored on the service level between service calls. In addition, enterprise services for the mySAP Business Suite do not store transient context information on the application level either. Hence, one service call only influences subsequent service calls by means of the persistent data in the database.

Furthermore, these enterprise services for the mySAP Business Suite are defined as synchronous services. This means, in particular, that they can be used without the SAP Exchange Infrastructure (SAP XI).

These enterprise services can be combined to form new business processes (service orchestration). User interfaces as well as composite applications can be built upon them.

SAP's Enterprise Service-Oriented Architecture (SOA)

SAP has been able to move the mySAP Business Suite toward enterprise SOA firstly by realizing the technical prerequisites for Web services and secondly, by introducing enterprise services.

1. *Generated Web Services*

   By means of SAP Web Application Server (SAP Web AS), SAP has supported the concept of Web services since SAP Web AS 6.40.

   SAP Web AS is the basis for all SAP applications. It comprises a framework for processing ABAP and Java programs and basic functionality for communications, transaction handling, database access, and other functions. The conventional way of calling SAP Web AS from other applications is via Business Application Programming Interfaces (BAPIs) or Remote Function Calls (RFCs). Both are proprietary SAP methods for inter-application communication.

   In SAP Web AS 6.40, SAP Web AS can act as a Web service provider, and there is a way of generating Web services based on any BAPI or RFC function.

   These generated Web services conform to the technical requirements of a Web service. However, their semantics are still those of the underlying BAPIs or RFCs. Hence, from a semantic point of view, they are not optimized for SOA.

2. *Enterprise Services*

   In order to take the step towards SOA on a semantic level as well, SAP introduced the notion of enterprise services. SOA based on enterprise services together with the required infrastructure is called enterprise SOA.

Prerequisites

To transform the mySAP Business Suite into an enterprise SOA solution, two main objectives have to be accomplished.

1. A technical framework has to be set up to allow service publication and the processing of service calls.

   To allow for as much reuse as possible and to avoid re-implementation of the business functionality of the mySAP Business Suite, the technical framework has been realized by means of a service interface layer on top of the mySAP Business Suite, as shown in the figure below.

2. The functionality of the mySAP Business Suite has to be optimized for use in a service-oriented application

The second task is to provide the business content of the mySAP Business Suite in a form that meets the principles of enterprise SOA. Here, SAP follows an outside-in, model-based approach. All services, and the business objects that these services operate on, are modeled according to current business practice.

mySAP Service Provisioning: Runtime

This section describes the concepts required to understand the execution of a Web service offered by the mySAP Business Suite.

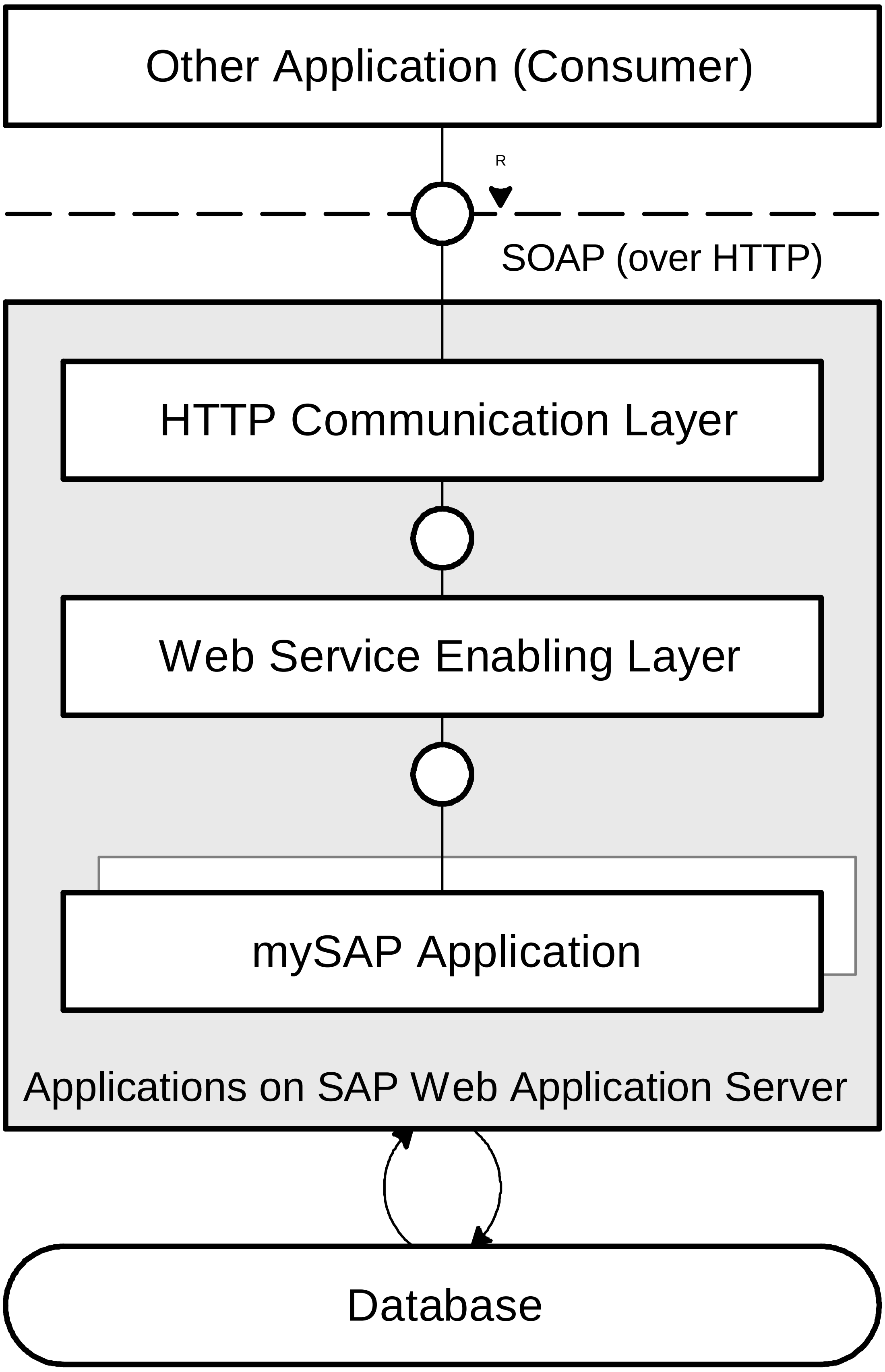
- The Runtime Architecture Overview describes the components of SAP Web AS that are involved in the execution of a Web service. It focuses only on the point-to-point scenario of executing Web services. In this scenario, a Web service offered by a provider is directly called by a consumer.
- The following specific concepts related to the invocation of synchronous Web services are also described:
  - mySAP as Service Provider and Service Consumer
  - Data Conversion and Custom Modifications
  - Data Access
  - Exceptions and Errors
  - Security

  Some of these concepts are not necessary to understand the services contained in the enterprise services packages, but they provide a complete picture of what service provisioning means at SAP.
- Web Service Invocation via XI Message Broker: An alternative to the point-to-point scenario is to use the SAP Exchange Infrastructure (XI) integration server as a message broker between the consumer and the provider. XI is not required for executing the synchronous Web services offered in the enterprise services packages, but only for asynchronous services. This scenario is therefore described separately.

Runtime Architecture Overview

The mySAP Business Suite is a family of applications running on SAP Web AS. In order to call the Web services offered by the mySAP Business Suite, another application acting as service consumer sends SOAP requests to SAP Web Application Server, which acts as a service provider. The following figure shows an overview of such a system.

Web Service-Enabled mySAP Business Suite


Other Application (Consumer)
R
SOAP (over HTTP)
HTTP Communication Layer
Web Service Enabling Layer
mySAP Application
Applications on SAP Web Application Server
Database

The applications which form the mySAP Business Suite can be distributed across different instances of SAP Web AS. This fact is omitted in the described architecture, as this bluebook does not focus on the implementation of SAP Web AS. It only describes components as necessary, to explain the execution of a Web service.

Structure

In order to allow the execution of Web services, two layers of functionality in SAP Web AS are required as shown above:

- The HTTP Communication Layer
- The Web Service Enabling Layer

Process

The majority of Web services provided by the enterprise services packages are invoked synchronously.

For more information on this process, see Invocation and Execution of Synchronous Web Services.

Note

The enterprise services packages for the mySAP Business Suite on SAP NetWeaver 2004s focus on providing synchronous Web services. However, Web services provided or called by an mySAP application need not be invoked synchronously. The underlying technical infrastructure of SAP Web AS also supports asynchronous service invocations. However, in order to use them reasonably, a networking infrastructure providing reliable messaging is required (see Web Service Invocation via XI Message Broker).

End of the note.

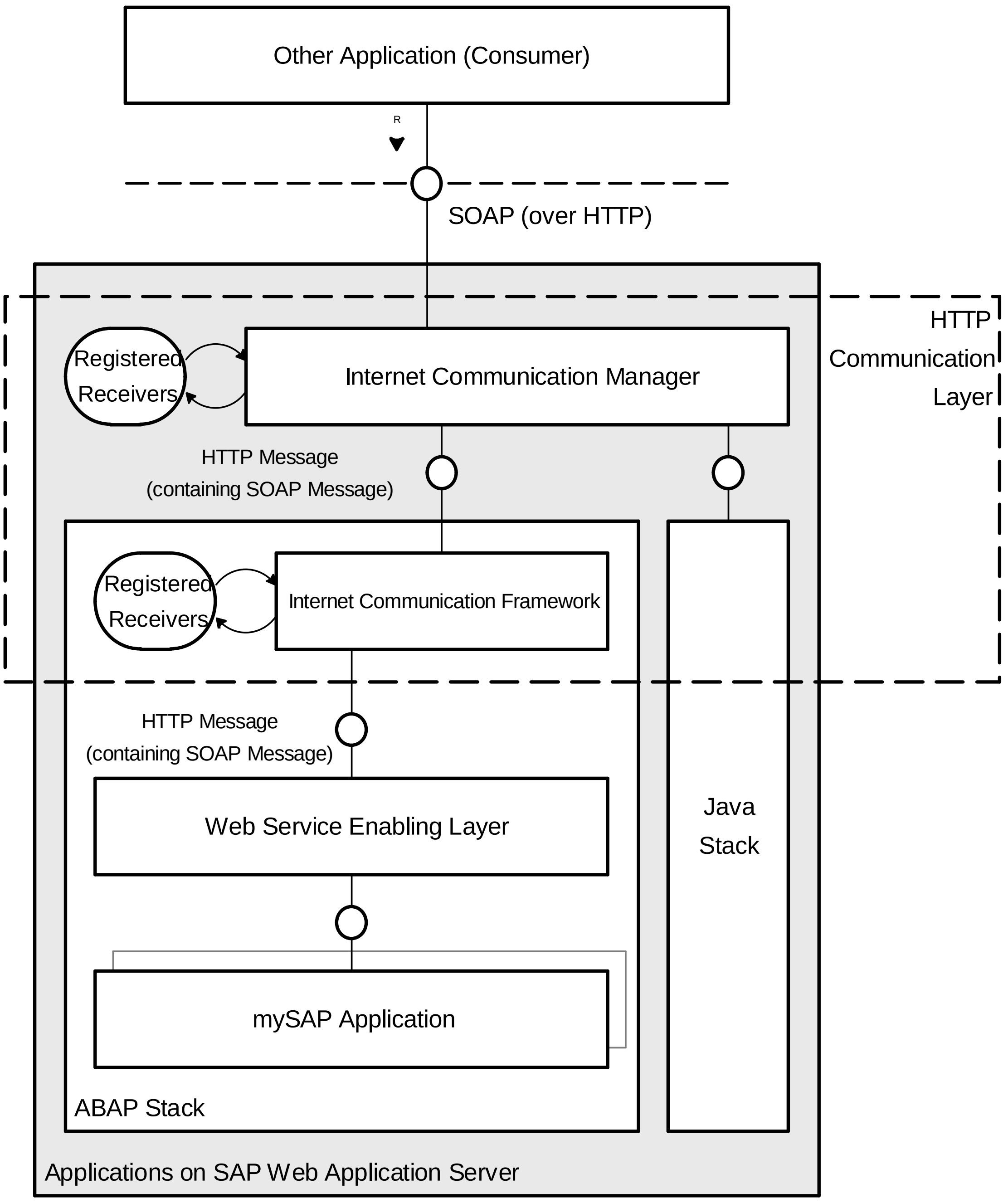
HTTP Communication Layer

The HTTP communication layer is one of two layers of functionality in SAP Web AS required for the execution of Web services.

The HTTP communication layer of the SAP Web AS handles all HTTP-based communication, which includes Web service communication. SAP Web AS can, for example, also act as a regular Web server, providing HTML Web pages. This layer is necessary for Web service invocations, because currently at SAP, all SOAP messages are sent via HTTP.

The main task of the layer is to accept incoming HTTP requests and dispatch them to the appropriate receiver within SAP Web AS. Dispatching is performed in two consecutive steps by the Internet Communication Manager and the Internet Communication Framework.

HTTP Communication Layer


Other Application (Consumer)
R
SOAP (over HTTP)
HTTP Communication Layer
Registered Receivers
Internet Communication Manager
HTTP Message (containing SOAP Message)
Registered Receivers
Internet Communication Framework
HTTP Message (containing SOAP Message)
Web Service Enabling Layer
mySAP Application
ABAP Stack
Java Stack
Applications on SAP Web Application Server

Note

According to the W3C recommendation, SOAP does not necessarily need to be based on HTTP, but can use other transport protocols, such as SMTP. However, as transport protocols other than HTTP are rarely used, SAP Web AS only supports SOAP communication via HTTP or HTTPS.

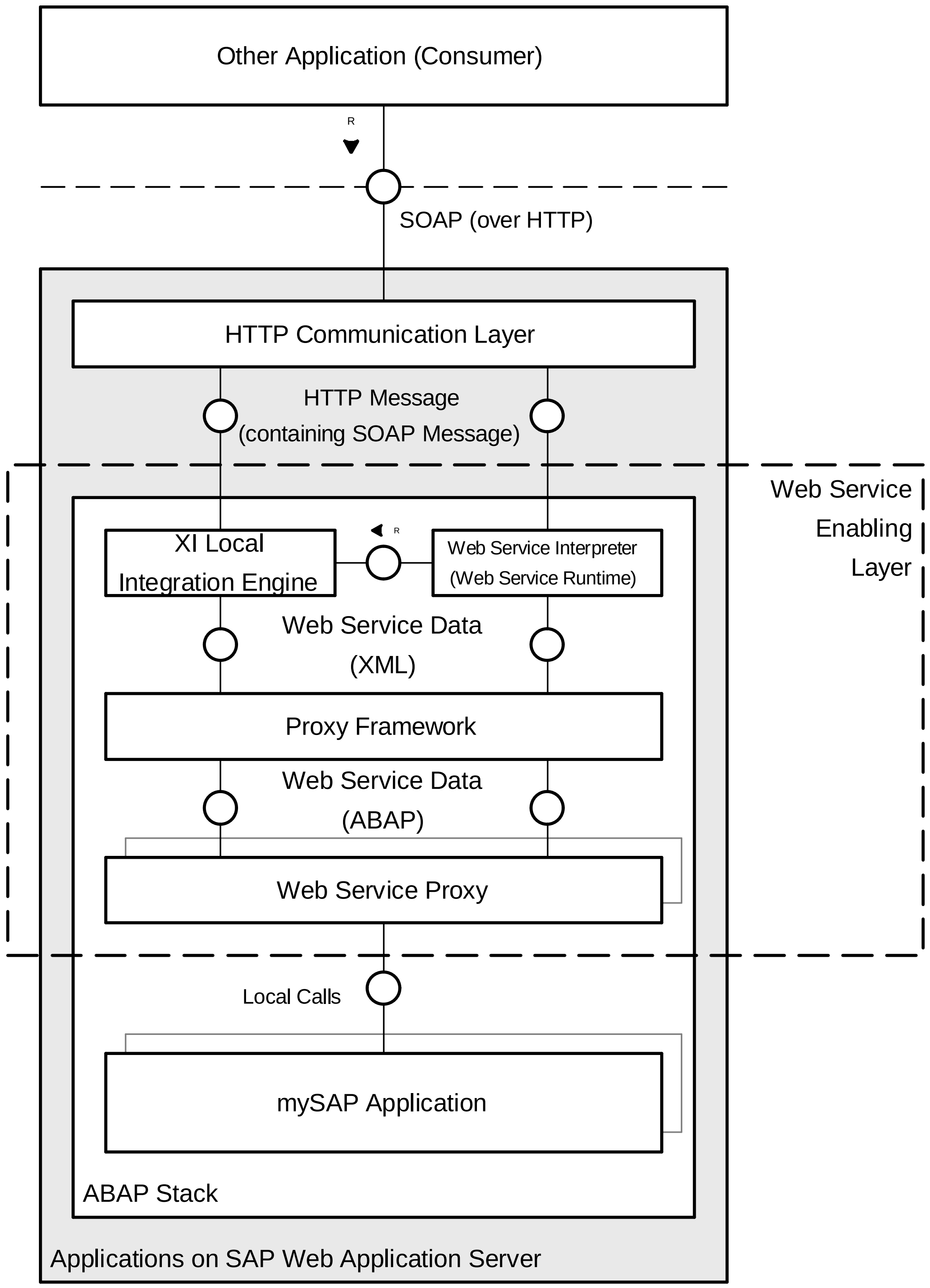
End of the note.

Web Service Enabling Layer

The Web Service Enabling Layer is one of two layers of functionality in SAP Web AS required for the execution of Web services.

The Web Service Enabling Layer receives HTTP-based SOAP messages from the HTTP Communication Layer. It interprets the SOAP messages and extracts the parameters for calling a Web service. The layer also contains the Web service *proxies* (as seen in the figure below), which are the implementation components that implement the Web service interface by forwarding calls to existing mySAP applications. In addition they can provide new functionality.

Web Service Enabling Layer



In addition to the Web service interpreter for interpreting point-to-point Web service calls, the figure above also depicts the *XI Local Integration Engine*. It is the counterpart of the Web service interpreter if Web services are executed via the XI infrastructure (see Web Service Invocation via XI Message Broker). However, it is also used in the point-to-point scenario, as it provides some functionality, such as logging, that is used by the Web service interpreter.

The Web Service Enabling Layer fulfills two tasks:

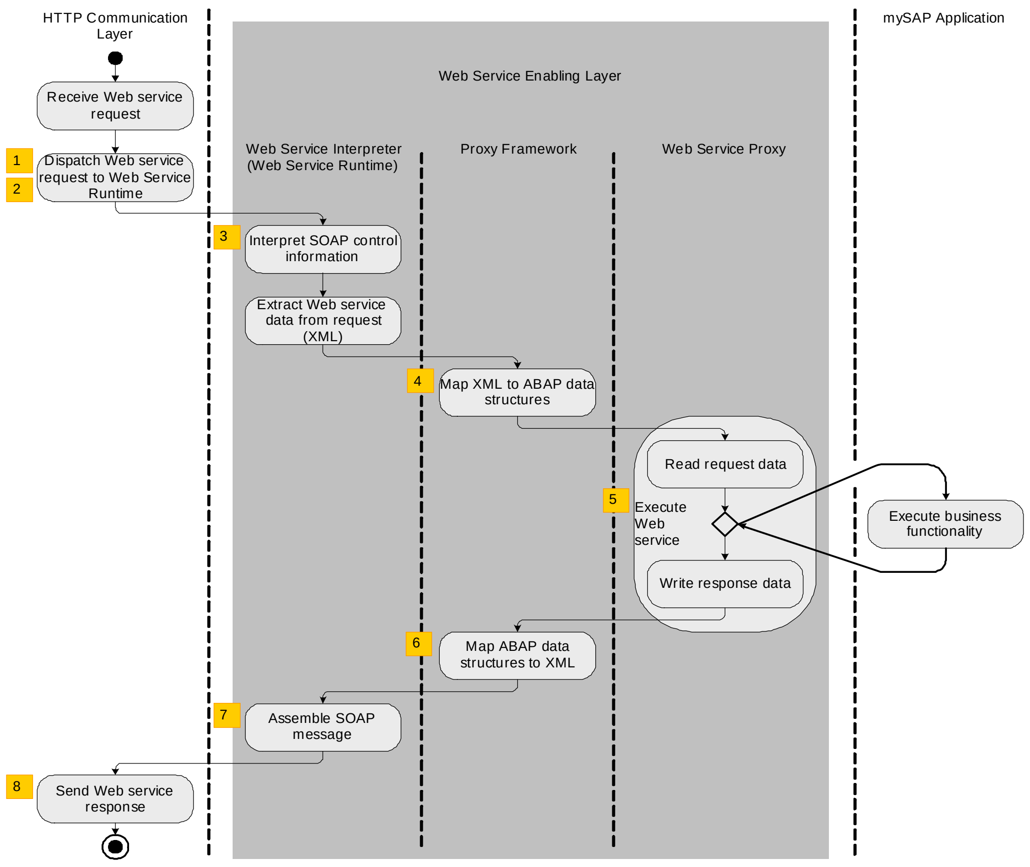
- Interpreting the SOAP messages and mapping the contained Web service data to a format that can be processed by existing applications on the SAP Web AS.
- Providing the Web services, meaning that this layer contains the implementations of the Web service interfaces. Such an implementation builds on existing functionality provided by mySAP applications.

Invocation and Execution of Synchronous Web Services

The majority of Web services provided by the enterprise services packages are invoked synchronously.

The following figure shows the execution of a synchronous web service. For the sake of clarity, the Web service consumer and the components of the HTTP Communication Layer are omitted.

Invocation and Execution of Synchronous Web Services


HTTP Communication Layer
Web Service Enabling Layer
mySAP Application
Receive Web service request
1
2
Dispatch Web service request to Web Service Runtime
Web Service Interpreter (Web Service Runtime)
Proxy Framework
Web Service Proxy
3
Interpret SOAP control information
Extract Web service data from request (XML)
4
Map XML to ABAP data structures
Read request data
5
Execute Web service
Execute business functionality
Write response data
6
Map ABAP data structures to XML
7
Assemble SOAP message
8
Send Web service response

Process

The execution of such a service takes eight steps. Steps one to five deal with dispatching the incoming message, mapping the contained data, and triggering the appropriate Web service proxy. The remaining steps are required to return the Web service result to the service consumer.

- Dispatching a request (steps 1–3)
- Processing a request (steps 4–5)
- Sending a response (steps 6–8)

The steps described are processed in every invocation of a Web service offered by the enterprise services packages.

Dispatching a Request

1. The Internet Communication Manager (ICM) decides whether to forward an incoming request to the ABAP stack or the Java stack. All services contained in the enterprise services packages are implemented in ABAP, which is why a request for the execution of such an enterprise service is forwarded to the ABAP stack
2. The Internet Communication Framework (ICF) receives all HTTP requests which are sent to the ABAP stack and forwards the request to the appropriate application. SOAP requests for the execution of Web services are forwarded to the Web Service Enabling Layer.

   The decisions made in both steps are based on the URL which is in the request. Each receiver is registered for a specific URL, or for a part of a URL, to specify the type of requests that it can accept.

   To determine whether the ABAP stack or the Java stack is addressed, the Internet Communication Manager analyzes the first part of the URL, which identifies the virtual host, for example https://www.myCompany.com:1234/WebServices/myService. The Internet Communication Framework analyzes the rest of the URL, for example https://www.myCompany.com:1234/WebServices/myService, to determine the application that is to receive the request.

   The HTTP Communication Layer does not modify the HTTP messages it forwards.

Processing a Request

All incoming Web service requests are processed by the components of the Web Service Enabling Layer in three sequential steps.

1. The Web service interpreter reads the SOAP message provided by the HTTP Communication Layer. It reads the information in the SOAP header and extracts the Web service data from the body of the message. The SOAP header is used to extend the standard SOAP protocol with information required for additional protocol services, such as reliable messaging. The Web service interpreter component, which is called Web service runtime in SAP Web AS, must interpret this information in order to provide the additional services.

2. The extracted XML Web service data is forwarded to the proxy framework. The proxy framework maps the data from XML to ABAP data structures and invokes the appropriate Web service proxy.

3. The Web service proxy provides functionality of the Web service by placing local calls to function modules, BAPIs and RFCs of mySAP applications. As a Web service proxy is a standard ABAP class, it can also contain further coding that adds to existing mySAP functionality.

Note

- Web service proxies can be used for both point-to-point and XI-based invocation of Web services.

- The Web service proxy described above is only one of the proxy types available in SAP Web AS and is the proxy for synchronous Web services in the inbound scenarios. For more information on all the proxy types available see Proxy Type.

End of the note.

Sending a Response

The steps required to send a response are as follows:

1. When the processing of the Web service proxy is finished, the proxy triggers the proxy framework to map the response data from ABAP data structures to XML.

2. The SOAP message containing the Web service response is assembled by the Web service interpreter, meaning that it adds SOAP control information to the header of the message and Web service data to the body of the message.

3. The HTTP Communication Layer sends the response to the specified receiver as an HTTP message.

mySAP as Service Provider and Service Consumer

Web service-based communication in which the mySAP application acts as a service provider, is referred to as an *inbound scenario*. Communication in which the mySAP application acts as the service consumer is referred to as an *outbound scenario*.

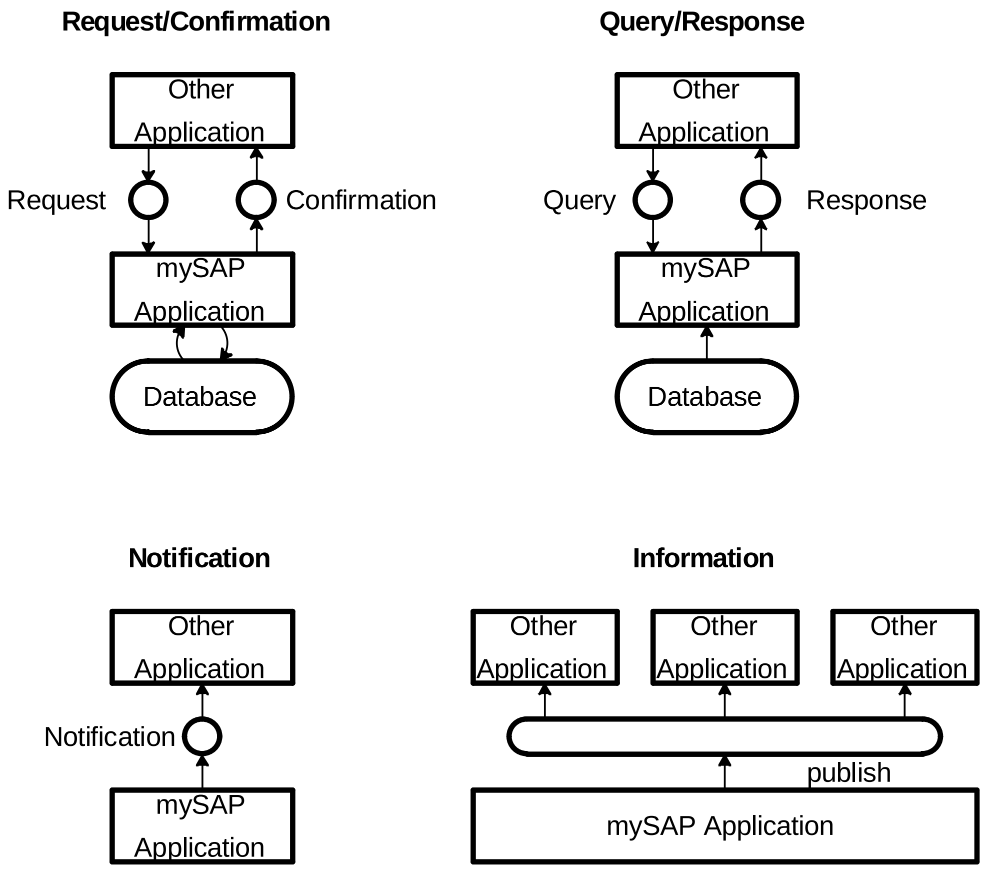
The communication patterns that can be used in these scenarios are described in the section Communication Pattern.

The Web service proxy described in the previous sections is only one of the proxy types available in SAP Web AS and is the proxy for synchronous Web services in the inbound scenarios. All the proxy types that are available are described in the section Proxy Type.

Communication Pattern

A communication pattern defines the choreography of a single interaction between the communication partners and defines which messages types are exchanged as request and response. Four patterns for communication between a mySAP application and another application have been implemented, as can be seen in the following figure.

Communication Patterns



The patterns can be categorized into patterns for transactional processing of data and patterns for data retrieval or information syndication. Patterns of the first category are used to change data of mySAP applications, while patterns of the second category are only used for retrieving data or information.

Transactional Data Processing

- *Request/Confirmation*: The basic pattern for transactional processing is called request/confirmation. It assumes that a Web service request is sent from another application to a mySAP application. The confirmation of the execution of the Web service is returned either immediately (synchronously) or later (asynchronously).

- *Notification*: A second pattern for transactional processing is called notification. Here, the other application sends a notification message to the mySAP application. This message leads to data changes. In contrast to the request/confirmation pattern, no response is sent.

Data Retrieval and Information Syndication

- *Query/Response*: The basic data provisioning pattern is called query/response. Another application requests some data from a mySAP application in a read-only form, specified by a query. Query/response communication can be either synchronous or asynchronous.

- *Information*: A second data provisioning pattern is the information pattern, which informs other registered applications of data changes in a mySAP application. The difference to the notification pattern is that the information message is typically sent to a list of subscribed receivers, and the receivers do not take immediate action upon the information received. No response to the information message is expected.

All communication patterns can be used in the outbound scenario as well. Communication patterns are realized by invoking Web services. Therefore, an application which is a consumer (or provider respectively) in a communication pattern need not always be a consumer on the level of Web service invocations but may also provide Web services that are required for the communication pattern (see figure).

The example in the above figure shows the asynchronous realization of a query/response communication pattern. On communication pattern level, a consumer sends a query to a provider and receives a response. On the Web service invocation level both communication partner consume and provide Web service.

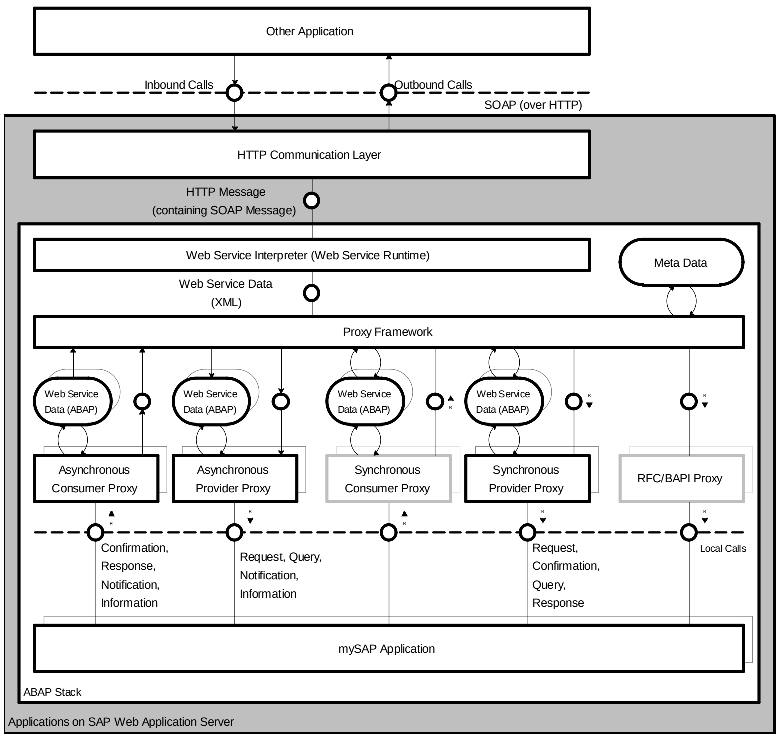
To distinguish between the two levels, in this document *inbound (or outbound) scenario* refers to the *communication pattern level* and *inbound (or outbound) call* refers to the *Web service invocation level*. An inbound (or outbound) scenario may comprise inbound and outbound calls.

Proxy Type

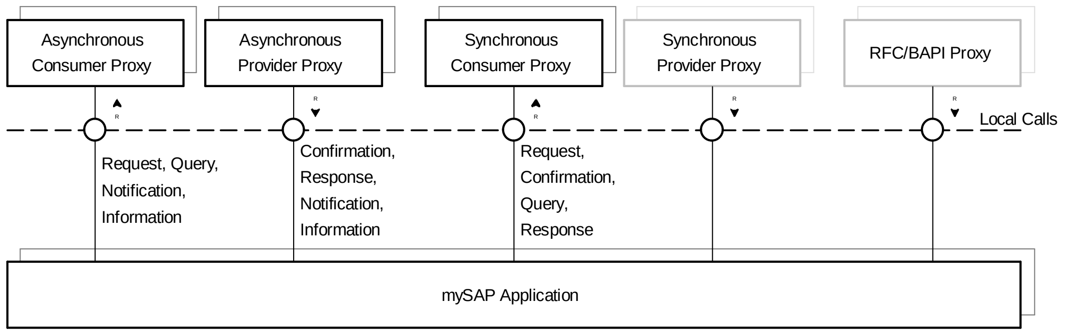
A Web service proxy, or proxy, is the implementation of a Web service interface in the Web Service Enabling Layer. Different types of proxies are used to implement the communication patterns, introduced in the previous section, in the inbound and outbound scenarios.

The first figure illustrates the proxy types used in the inbound scenario. The second figure shows a slightly modified version of the lower part of the first. It depicts the proxies used in the outbound scenario.

Proxy Types Used in the Inbound Scenario


Other Application
Inbound Calls
Outbound Calls
SOAP (over HTTP)
HTTP Communication Layer
HTTP Message (containing SOAP Message)
Web Service Interpreter (Web Service Runtime)
Meta Data
Web Service Data (XML)
Proxy Framework
Web Service Data (ABAP)
Web Service Data (ABAP)
Web Service Data (ABAP)
Web Service Data (ABAP)
Asynchronous Consumer Proxy
Asynchronous Provider Proxy
Synchronous Consumer Proxy
Synchronous Provider Proxy
RFC/BAPI Proxy
Confirmation, Response, Notification, Information
Request, Query, Notification, Information
Request, Confirmation, Query, Response
Local Calls
mySAP Application
ABAP Stack
Applications on SAP Web Application Server

Proxy Types Used in the Outbound Scenario


Asynchronous Consumer Proxy
Asynchronous Provider Proxy
Synchronous Consumer Proxy
Synchronous Provider Proxy
RFC/BAPI Proxy
Local Calls
Request, Query, Notification, Information
Confirmation, Response, Notification, Information
Request, Confirmation, Query, Response
mySAP Application

Note

The names of the proxy types used in this document are the names used in SAP NetWeaver NewYork. In SAP NetWeaver 2004s the proxy types are called client (consumer) and server (provider) proxy.

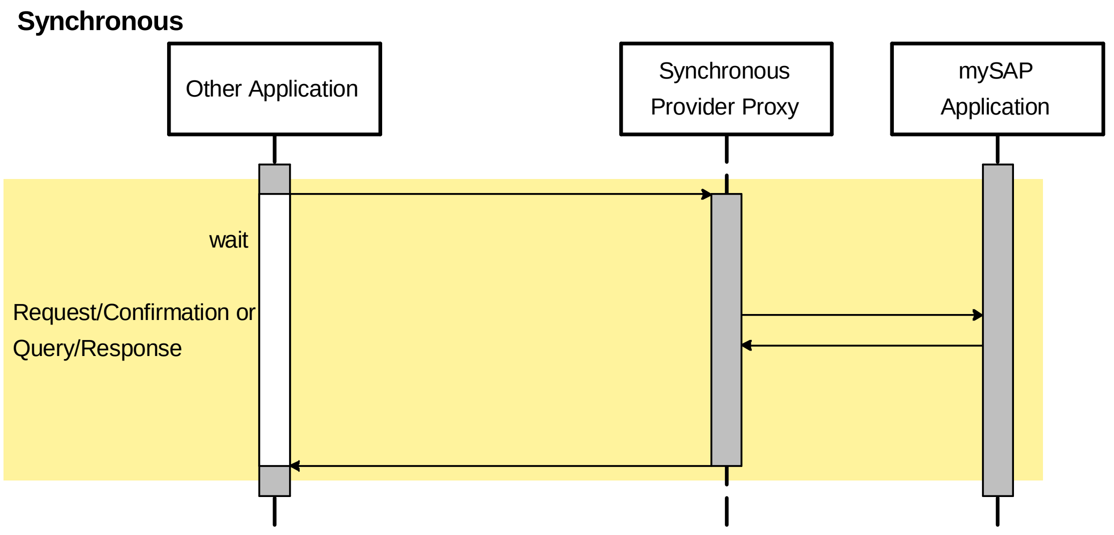
End of the note.

Proxy Type	**Proxy**
Asynchronous Proxies	These proxy types send or receive asynchronous Web service calls. • Asynchronous Consumer Proxy: This proxy type sends asynchronous Web service calls to a provider. • Asynchronous Provider Proxy: This proxy type receives asynchronous Web service calls. The Web services provided by the proxy are based on the functionality of existing applications within SAP Web AS.
Synchronous Proxies	These proxy types send and receive synchronous Web service calls. • Synchronous Consumer Proxy: This type of proxy sends synchronous Web service calls to a provider and waits for the response. • Synchronous Provider Proxy: This type of proxy receives synchronous Web service calls and sends a response to the consumer once processing has finished.
2 RFC/BAPI Proxies	This proxy type wraps an existing RFC/BAPI or function group as Web service. Therefore, an RFC/BAPI proxy also is a synchronous provider proxy. However, this proxy is depicted separately in Figure 2–7 Proxy Types used in the Inbound Scenario Note RFC/BAPI proxies are generated automatically without any further need of implementation effort. This is why they are not visible in the development environment. However, it is necessary to define a virtual interface for these proxies. This allows to hide specific parameters of an RFC/BAPI call in a Web service. It is, for example, possible to set default values for parameters which should not be available in the Web service. End of the note.

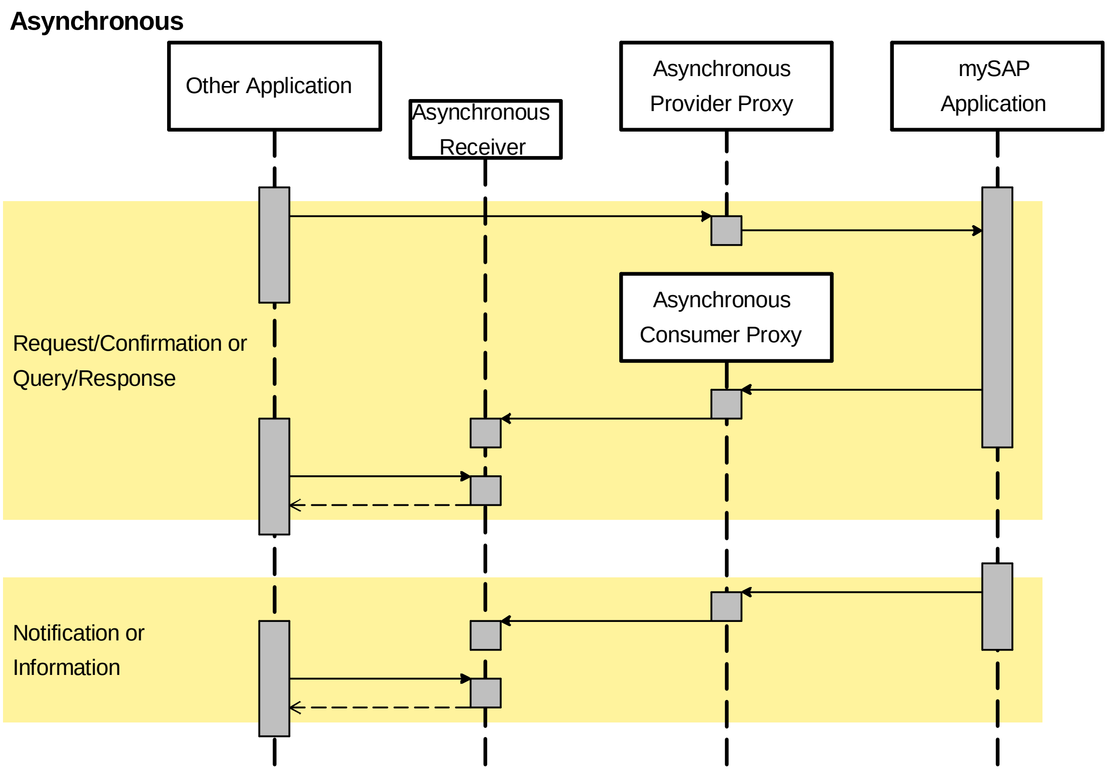
In the two figures above, the messages types (see Communication Pattern) that can be sent or received by each proxy type are shown next to the channels connecting the proxies and the mySAP applications. The usage of the proxies in the communication patterns are shown in the two figures below. All patterns, except the notification and information patterns, are depicted in the inbound scenario. If used in the opposite direction, the consumer/provider proxies have to be replaced with their counterparts.

The first figure below shows the communication patterns available in the synchronous scenario, while the second shows the communication patterns available in the asynchronous scenario.

Usage of Proxy Types in Synchronous Scenarios


Synchronous
Other Application
Synchronous Provider Proxy
mySAP Application
wait
Request/Confirmation or Query/Response

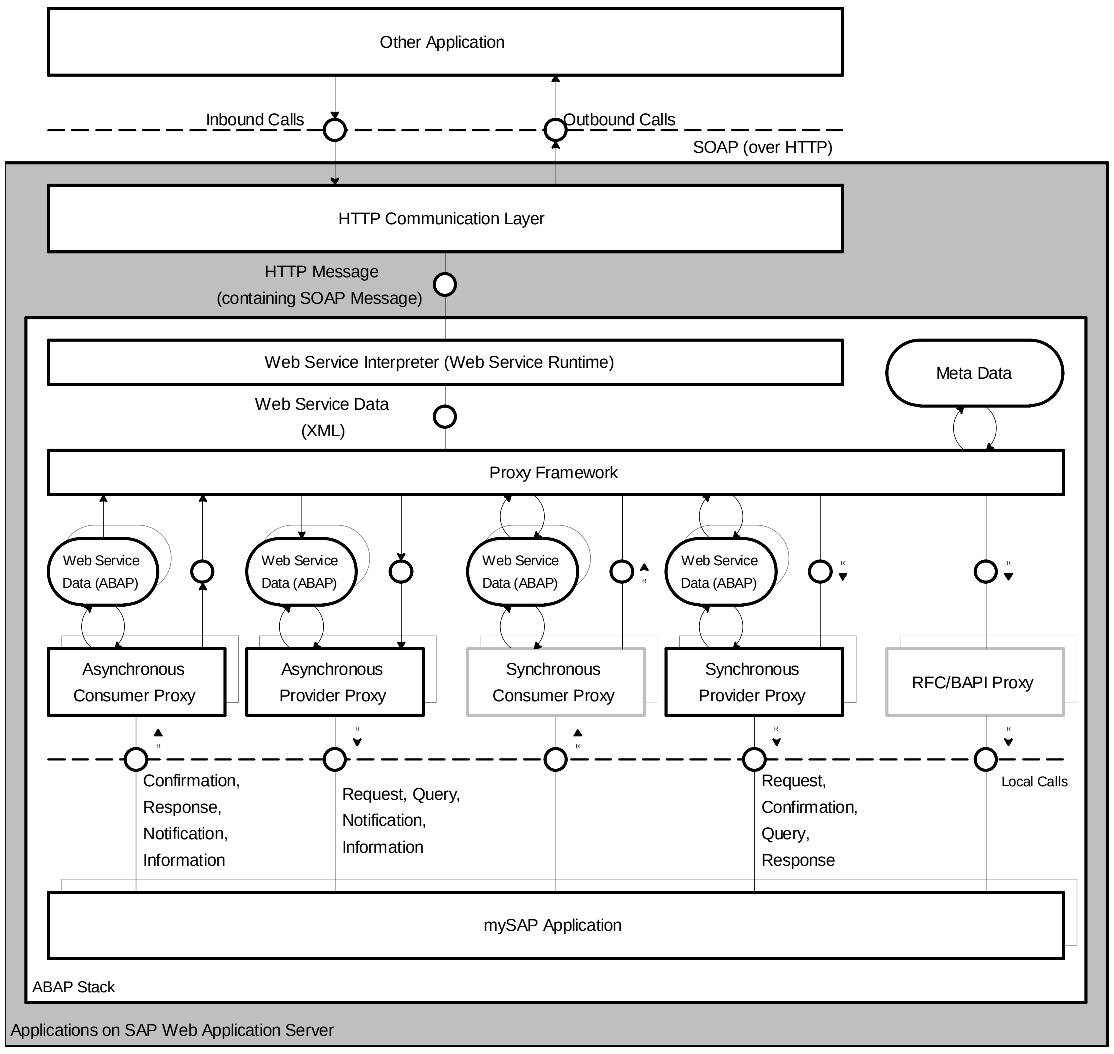
Use of Proxy Types in Asynchronous Scenarios


Asynchronous
Other Application
Asynchronous Receiver
Asynchronous Provider Proxy
mySAP Application
Asynchronous Consumer Proxy
Request/Confirmation or Query/Response
Notification or Information

Data Conversion and Customer Modifications

The data contained in a Web service call is represented by XML data types. To allow ABAP applications to process the data, it has to be mapped to ABAP data types. The ABAP data of a Web service call is depicted by the Web service data storages in the following figure.

Proxy Types Used in the Inbound Scenario


Other Application
Inbound Calls
Outbound Calls
SOAP (over HTTP)
HTTP Communication Layer
HTTP Message
(containing SOAP Message)
Web Service Interpreter (Web Service Runtime)
Meta Data
Web Service Data
(XML)
Proxy Framework
Web Service Data (ABAP)
Web Service Data (ABAP)
Web Service Data (ABAP)
Web Service Data (ABAP)
Asynchronous Consumer Proxy
Asynchronous Provider Proxy
Synchronous Consumer Proxy
Synchronous Provider Proxy
RFC/BAPI Proxy
Confirmation, Response, Notification, Information
Request, Query, Notification, Information
Request, Confirmation, Query, Response
Local Calls
mySAP Application
ABAP Stack
Applications on SAP Web Application Server

Depending on the type of proxy, these storages contain either a request or a response only, or both requests and responses. The ABAP data structures implementing these Web service data storages are provided by the proxies. The proxy framework knows how to map the XML data types to an ABAP representation and vice versa.

- Proxy Meta Data: The information required to do the mapping is contained in the proxy meta data.
- Application Level Mapping of Data Types: The technical mapping between XML and ABAP performed in the proxy framework is not sufficient. A conversion of data types on the application level is required for specific data types.

Proxy Meta Data

The data contained in a Web service call is represented by XML data types. To allow ABAP applications to process the data, it has to be mapped to ABAP data types. The information required to do the mapping is contained in the proxy meta data.

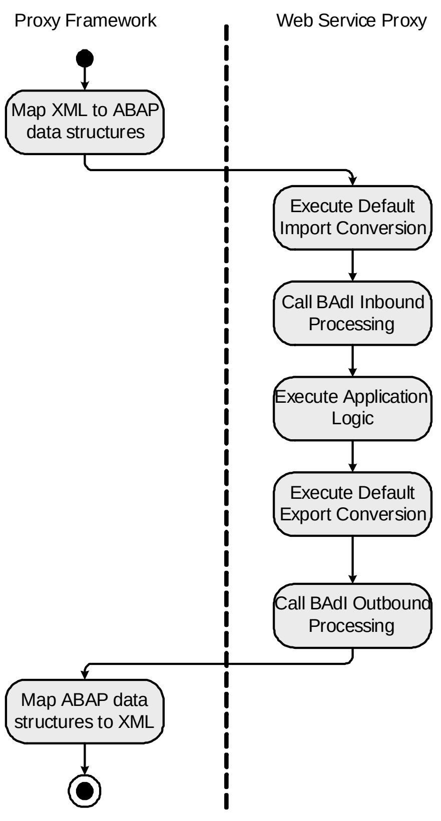
The proxy framework creates mapping programs, called *Simple Transformations*, based on the given mapping information specified for the data types. For incoming messages, a mapping program uses the XML parameters as input and fills the ABAP data structure. For outgoing messages, a different mapping program with ABAP as input and XML as output is executed. The information contained in the proxy meta data is derived from the abstract service definitions during design time (see mySAP Service Provisioning: Design Time.

Application Level Mapping of Data Types

The technical mapping between XML and ABAP performed in the proxy framework is not sufficient. A conversion of data types on the application level is required for specific data types, too. Examples of such data types are codes, units, and time-related data types. In a Web service call, such data types are typically specified in a standardized format, such as ISO codes. However, mySAP applications have a different format. Therefore, the Web service proxy has to convert the data types into the correct format before it can begin to execute the application logic.

Process

Data Conversion


Proxy Framework
Web Service Proxy
Map XML to ABAP data structures
Execute Default Import Conversion
Call BAdI Inbound Processing
Execute Application Logic
Execute Default Export Conversion
Call BAdI Outbound Processing
Map ABAP data structures to XML

Business Add-In (BadI)

As the data conversion on the application level also affects data types that can be extended by customers, such as code lists, a Business Add-In (BAdI) is provided for adding custom import and export data conversion (see figure).

In addition to data conversion alone, the methods for inbound and outbound processing provided by the BAdI, can be used by customers to modify the behavior of a Web service.

Example

Per default, a Web service that retrieves information about all goods that can be ordered from a supplier by a retailer does not filter the returned information. However, business cases exist, in which the returned information must be filtered according to certain criteria. If, for example, the retailer is a store, only information about those goods should be retrieved that are part of the retailer’s assortment of goods.

End of the example.

The BAdI for the import and export processing in Web service calls uses a sorter concept for the registered BAdIs implementations. Each implementation is categorized as SAP, partner or customer implementation. The implementations are executed in this order.

The BAdI methods can access and modify all parameters of a Web service, even parameters that have been added through custom extensions.

Note

The data structures implementing the Web service data (ABAP) storages in the figure *Proxy Types used in the Inbound Scenario* (see Proxy Type) are called *Data Type Proxies* in the development environment

End of the note.

Data Access

Consistency and correctness of the data in the database have to be ensured for all services in the mySAP Business Suite.

Services for the enterprise services packages for the mySAP Business Suite 2005 are stateless services. On the service level, no context information is stored on the provider side between service calls. This means that each service call is an atomic transaction. The execution of a single mySAP Business Suite service always transfers the database from one consistent state to another. It is not allowed for a service to finish with an inconsistent database state, requiring that further service calls will fix it. Therefore, it is not possible to have transactions on the database level that span several service calls.

The following consequences have to be considered:

- *Snapshot data*: Data retrieval and subsequent data modification cannot be combined in one transaction. Hence, data captured by stateless services is not real-time data. Since data is only locked as long as service processing takes place, the service consumer retrieves data as a snapshot taken at a certain point in time. For more information see Concurrency Control.

- *Reliable messaging*: Since the standard SOAP protocol does not provide a means for reliable messaging, protection against loss and multiple execution of web service requests has to be provided. For more information see Reliable Messaging.

- *Commit*: The work has to be committed to the database at the end of each service execution that involves data modification. For more information see Commit Procedures.

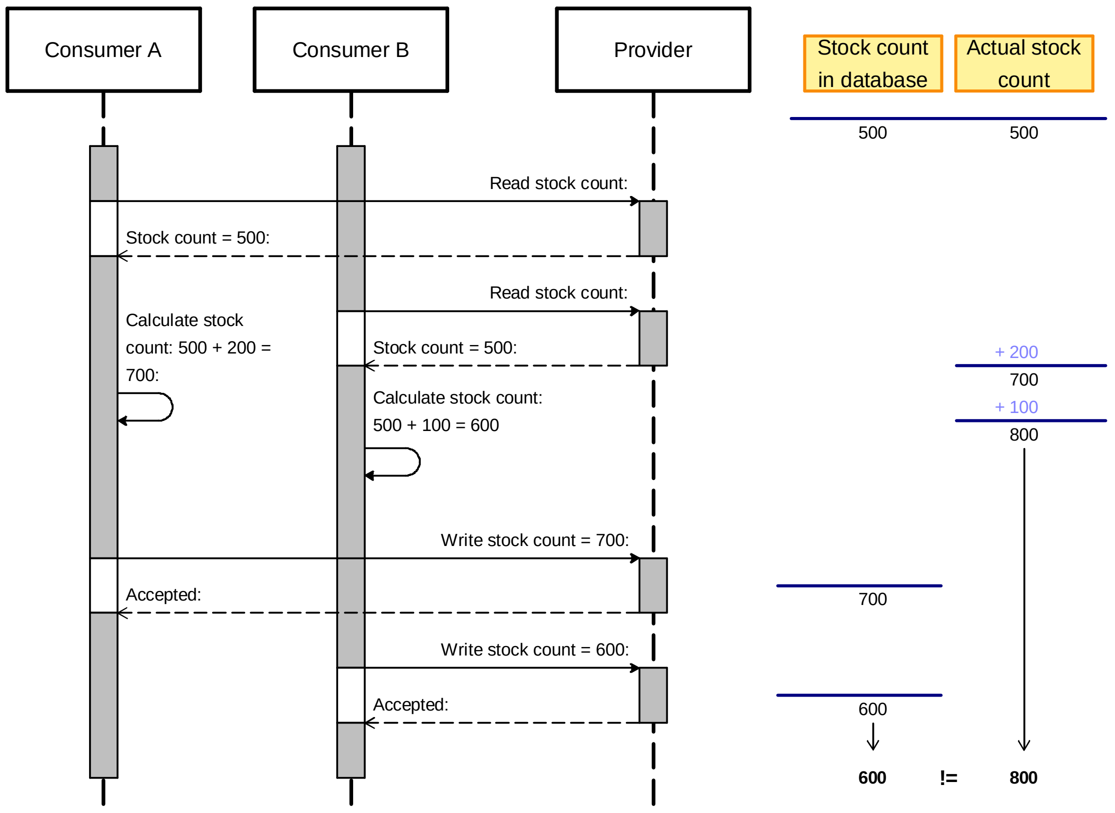
Concurrency Control

Each service in the enterprise services packages constitutes an atomic transaction. That is, a single service call always transfers the global data in the database from one consistent state to another.

However, though the database state is always consistent, it may still become semantically incorrect if multiple service consumers are interacting concurrently on a pair of services for retrieving or modifying a piece of data respectively.

Example

Overwrite Strategy Leading to Incorrect Database Content



Two consumer applications A and B are about to independently post the receipt of a delivery of screws. After querying the current quantity of screws in the warehouse, they calculate the new quantity of screws based on the current quantity and the delivery size. Subsequently they write the result in the sequence A-B.

If no precautions are taken, the new value in the database will be the amount calculated by application B, as B was the last one to write the value. Though there is no consistency problem, the value in the database is still not the right value from a business point of view.

End of the example.

To avoid such problems due to the interleaved execution of corresponding data retrieval and data modification Web services, it is necessary to implement adequate *prevention strategies*.

Basically, there are three different conciliation strategies which may be appropriate depending on the business semantics of the set of data in question. These are:

- Overwrite
- Delta Update
- Optimistic Locking

Overwrite

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.

In the example in the previous section *Concurrency Control*, this strategy is applied and leads to incorrect data. However, there are business cases for which this strategy is appropriate. An example is the recording of the stock count as determined during a physical inventory. It is desirable that the result of the last physical inventory, which is the most recent stock count, is always stored in the database

Services for which the last-one-wins strategy is appropriate are called change services. The term “change” indicates that the service overwrites the old data with new up-to-date data, ignoring the old data.

Delta Update

For the delta update strategy, also called *everybody-wins* strategy, each database modification attempt is accepted as well, but in contrast to the overwrite strategy, the data provided by different consumers is accumulated in the database.

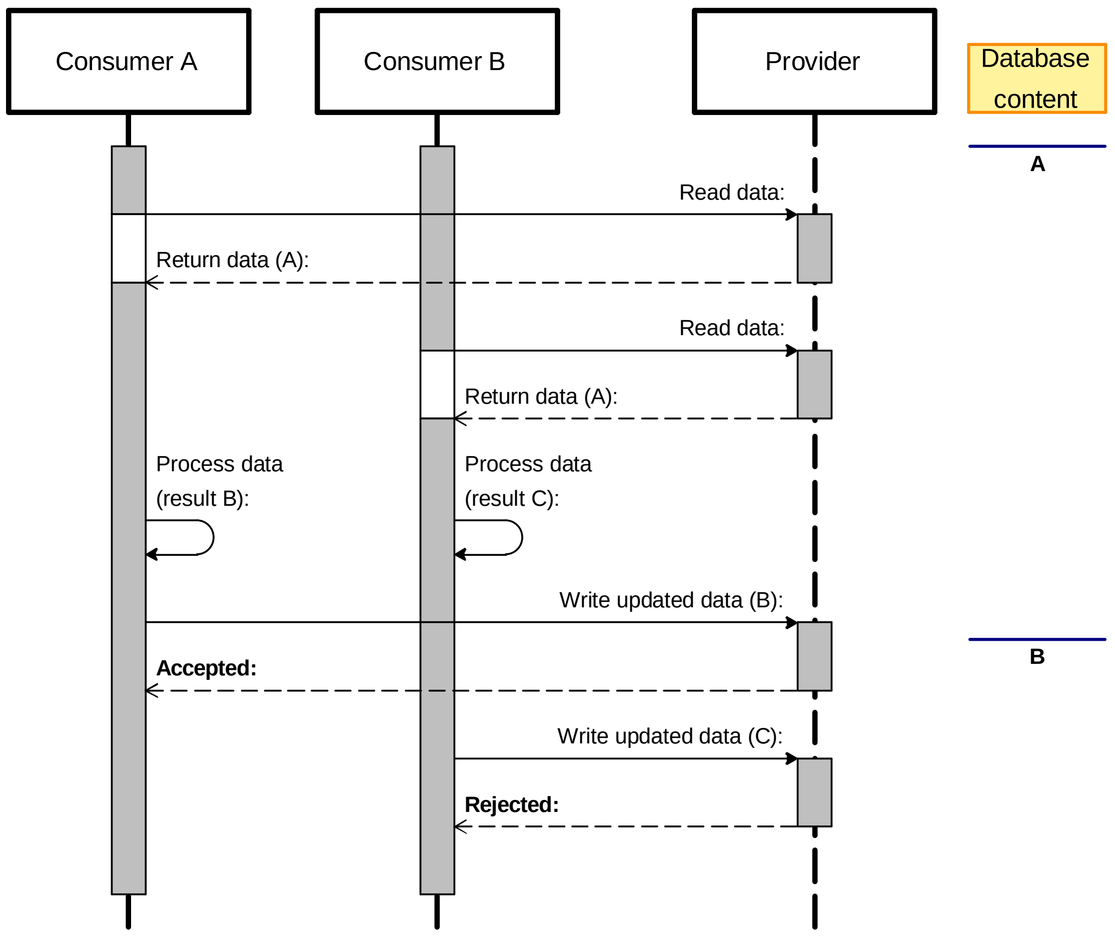
For instance, in the example in the previous section *Concurrency Control*, the applications could provide the changes to stock quantity instead of the new total stock count. All provided changes can then simply be added to the current stock count in the database with the database always containing the correct stock count.

The feasibility and implementation constraints of delta updates depend on the concrete business semantics of the service.

Optimistic Locking

For the optimistic locking strategy, also called *first-one-wins* strategy, only the first one of a sequence of consecutive database update attempts is successful, so the data provided in the first attempt will persist in the database, whereas further attempts fail. According to this strategy, the second database update attempt fails, and the second application is forced to query the current stock amount again and recalculate the new total stock amount.

Optimistic Locking Strategy



Update Services

The term optimistic locking refers to the fact that, in the first place, no locking actually takes place at all, assuming that in most cases no conflicts occur. Instead, for each modification attempt there is a check as to whether a conflict has occurred, and the modification attempt is rejected if need be.

Services for which the first-one-wins strategy is appropriate are called update services. The term “update” indicates that the service provides renewed data, the determination of which has been based upon the data read from the database before.

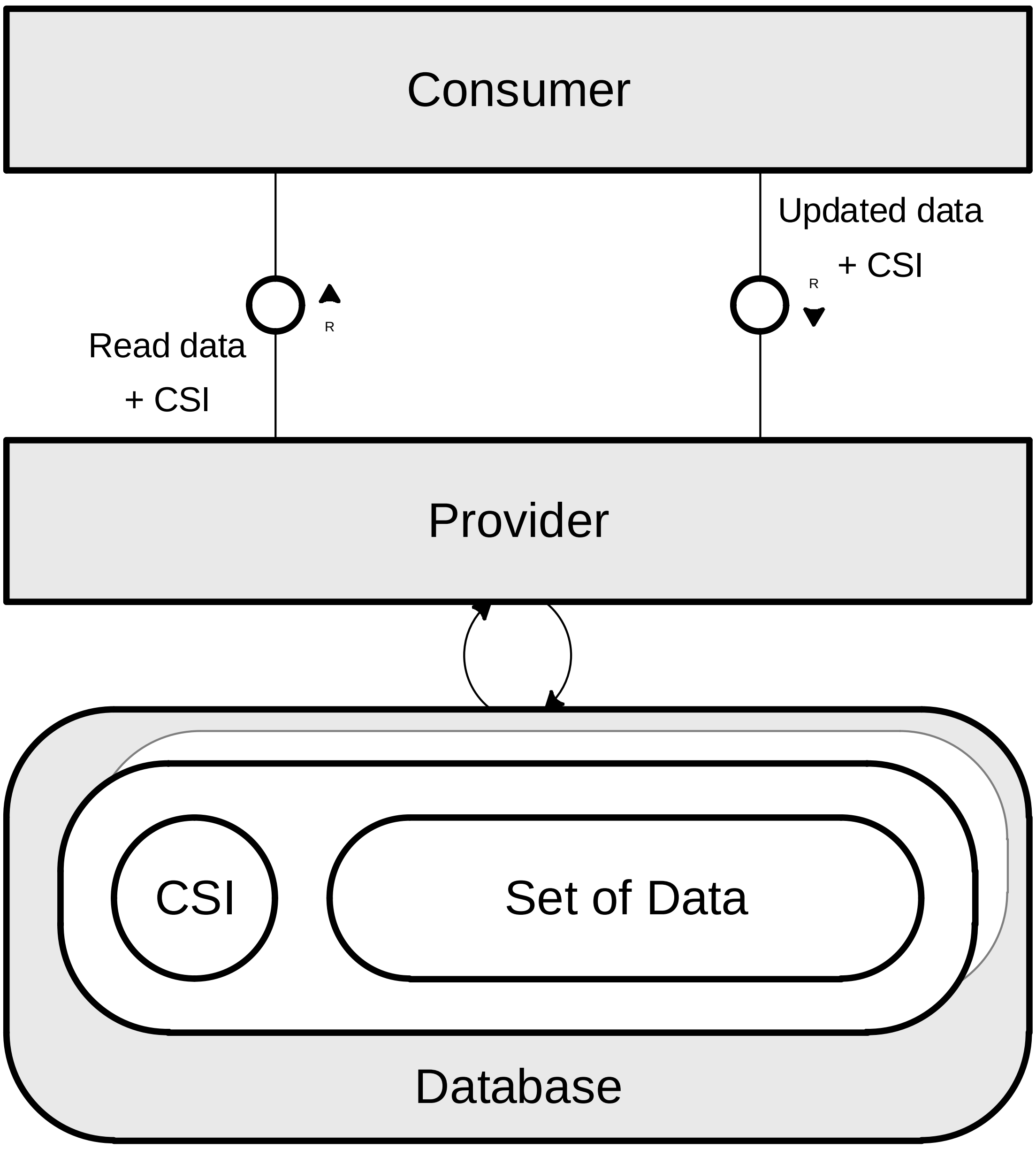
As explained above, the optimistic locking strategy is to ensure that, of all the service consumers which have read a certain database state, only the first one can commit its updated data. The same strategy can be expressed the other way round as well: Whenever a service consumer wants to store some updated data, it has to be ensured that the data it has read before is still the current data in the database. Otherwise it may have based its data processing on obsolete data.

Change State Identifier

To realize this, the service consumer must be able to indicate which data it has read originally. Therefore, an identifier is assigned to all pieces of data that are to be read, which uniquely identifies the state of the data. This identifier is called change state identifier (CSI). The change state identifier must be guaranteed to change with each database update.

As illustrated in the figure below, the consumer always receives the change state identifier together with the requested data. In a subsequent update request, the consumer can use the identifier to specify the state of the data which the update request refers to. If the specified state is no longer the current state, the update request is denied. In this case, the consumer has to read the new state and repeat the update request.

Change State Identifier



The sole requirement for a change state identifier is that it must change with each database update. For the enterprise services for the mySAP Business Suite, an existing database field will be used if an appropriate one is available (such as a change document number or the time stamp of the last document modification). Otherwise, a hash value generated from the current data is used as the change state identifier. As a hash value can be generated from the current data whenever needed, it does not need to be stored in the database. This is an advantage compared to other possible implementations such as an update counter or an update time stamp.

The consumer is not allowed to interpret the change state identifier. For the consumer, it is a token in the form of an arbitrary character string without semantics, because this allows for the underlying implementation to be changed later on.

Reliable Messaging

While SAP XI ensures reliable messaging for broker-based service calls via SAP XI (see Web Service Invocation via XI Message Broker), the standard SOAP protocol does not include any means of reliable messaging. Hence, in the point-to-point scenario, where Web services are called directly using the SOAP protocol, there is no guarantee that each request message sent by the consumer will arrive exactly once at the provider. It may get lost or may arrive several times in case of network problems. Even if the request message arrives at the provider, the response message may get lost during network transport as well. Hence the consumer might assume that its request did not arrive and re-send it.

Protection against loss and multiple execution of web service requests must therefore be provided. Three mechanisms are available. These are:

- Restartable Services
- Verification Services
- Idempotent Services

Restartable Services

It may be possible to implement some services in such a way that multiple invocations do not result in incorrect data in the database. For example, if a service that changes the address information of a customer is invoked several times with the same address as a parameter, this will still result in the correct address being stored in the database. Such services are called *restartable services* or said to have *restart capability*. In the case of restartable services, the consumer can repeat any request without causing the data to be incorrect. Change services are typically restartable.

For many services, however, it is crucial that the service is executed exactly once when requested. For example, if a new purchase order is to be created via a service invocation, it is neither acceptable to have no purchase order created nor to have two or more duplicates created.

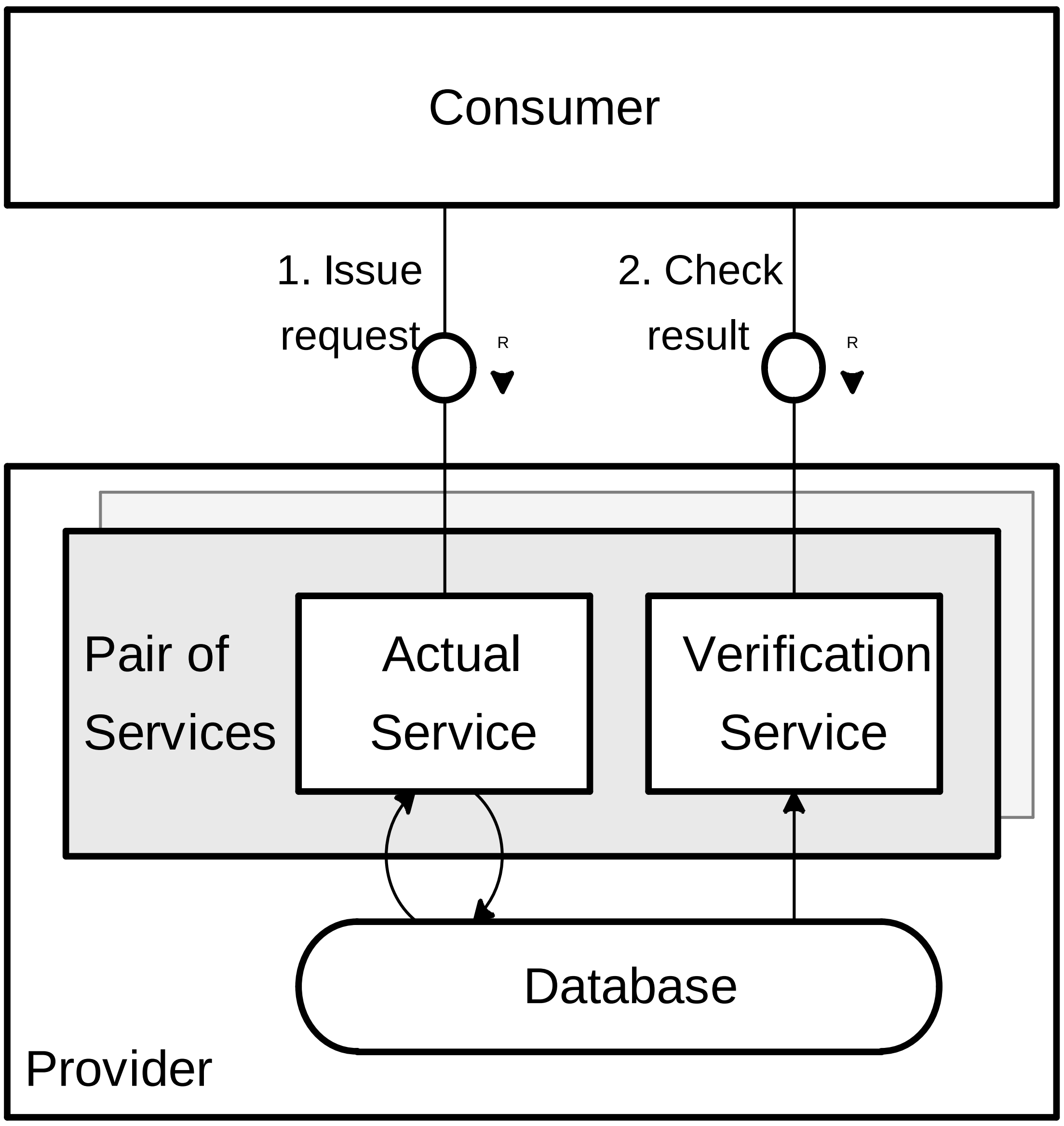
For these services, there are two mechanisms available for ensuring that a service is executed exactly once when it is requested by a consumer: Verification Services and Idempotent Services.

Verification Services

For many services, it is crucial that the service is executed exactly once when requested. For example, if a new purchase order is to be created via a service invocation, it is neither acceptable to have no purchase order created nor to have two or more duplicates created.

One mechanism, which can be used for the services in the enterprise services packages for the mySAP Business Suite 2005, provides the consumer with a way of ensuring that a service is executed exactly once when it is requested by a consumer. The idea is to create an additional service, a *verification service*, accompanying the actual service.

Verification Services



The verification service is a service that does not modify the database and has such semantics that it can be used to check whether a request to the actual service has been successful or not. In many cases, verification services do not need to be implemented explicitly, because an appropriate query service exists anyway.

In the example of a purchase order creation service, a verification service would retrieve purchase orders according to unique identifying parameters, such as a creation date within a certain time period and a specific creator.

Integration

This mechanism has two disadvantages:

- It may not be reasonable to create a matching verification service for every service.
- It requires the consumer to cope with up to twice the amount of services it actually needs, which implies some additional configuration effort.

Another mechanism for ensuring that a service is executed exactly once when it is requested by a consumer is provided by Idempotent Services.

Idempotent Services

For many services, it is crucial that the service is executed exactly once when requested. For example, if a new purchase order is to be created via a service invocation, it is neither acceptable to have no purchase order created nor to have two or more duplicates created.

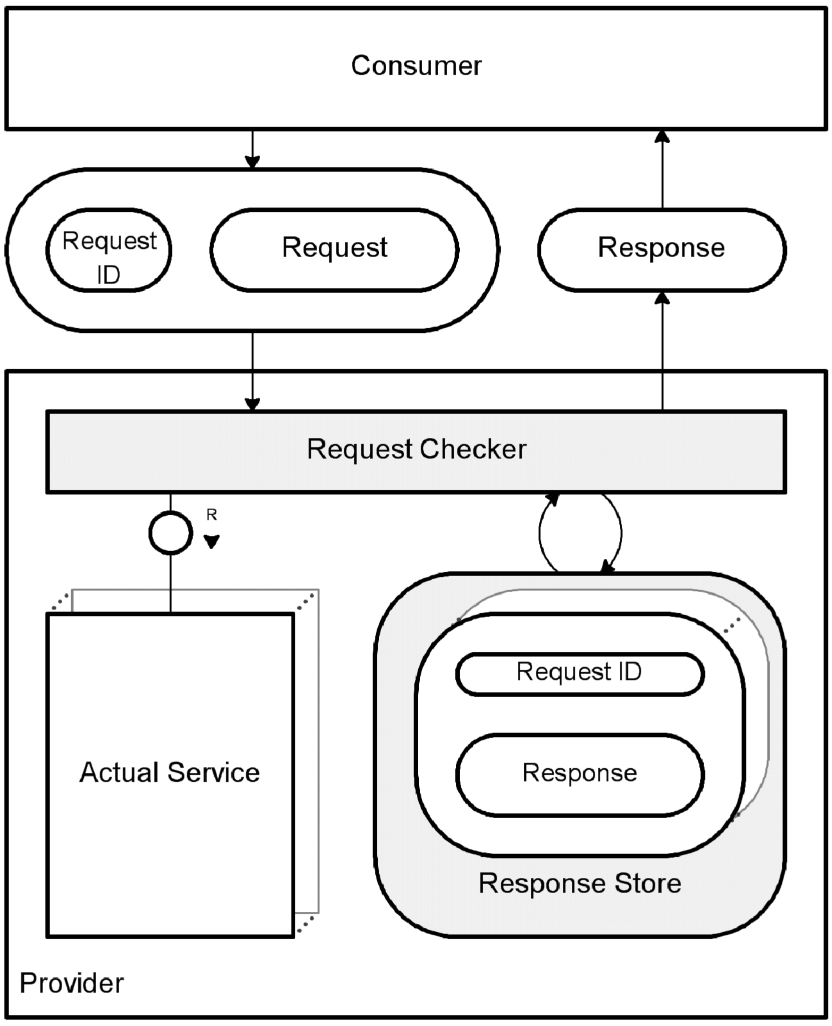
With Netweaver 2004S SPS09 / ECC SE 600 SP03, a framework is provided that ensures on the provider side that service calls are executed exactly once. Such services are called *idempotent services* or said to have *idempotency*.

The framework is used to store a message created in response to an incoming service call before the response is actually sent to the consumer. The response is identified by the request message's ID. For each incoming request, the provider can thus determine if the exact same request has already been processed by checking if a response for the request ID has been stored, and if yes directly return the earlier created response without really re-processing the request.

The figure below shows how a service provider implementation using the framework looks like conceptually ("Request ID" corresponds to the message ID).

The framework takes care of locking the response store, and of removing stored responses after a certain period such that the database isn't filled up unnecessarily.

Idempotent Services



Integration

See Important Facts for Service Implementers

Important Facts for Implementers

Implementation Considerations

The framework or the “request checker” as illustrated in the section Idempotent Services is not called automatically by the web service infrastructure or the local integration engine. Each proxy implementation that has to ensure exactly-once processing must actively call the framework.

You must manually configure the time periods governing for how long messages and information about sent messages are saved. You can find a description of how to do this in Configuring Idempotent Services.

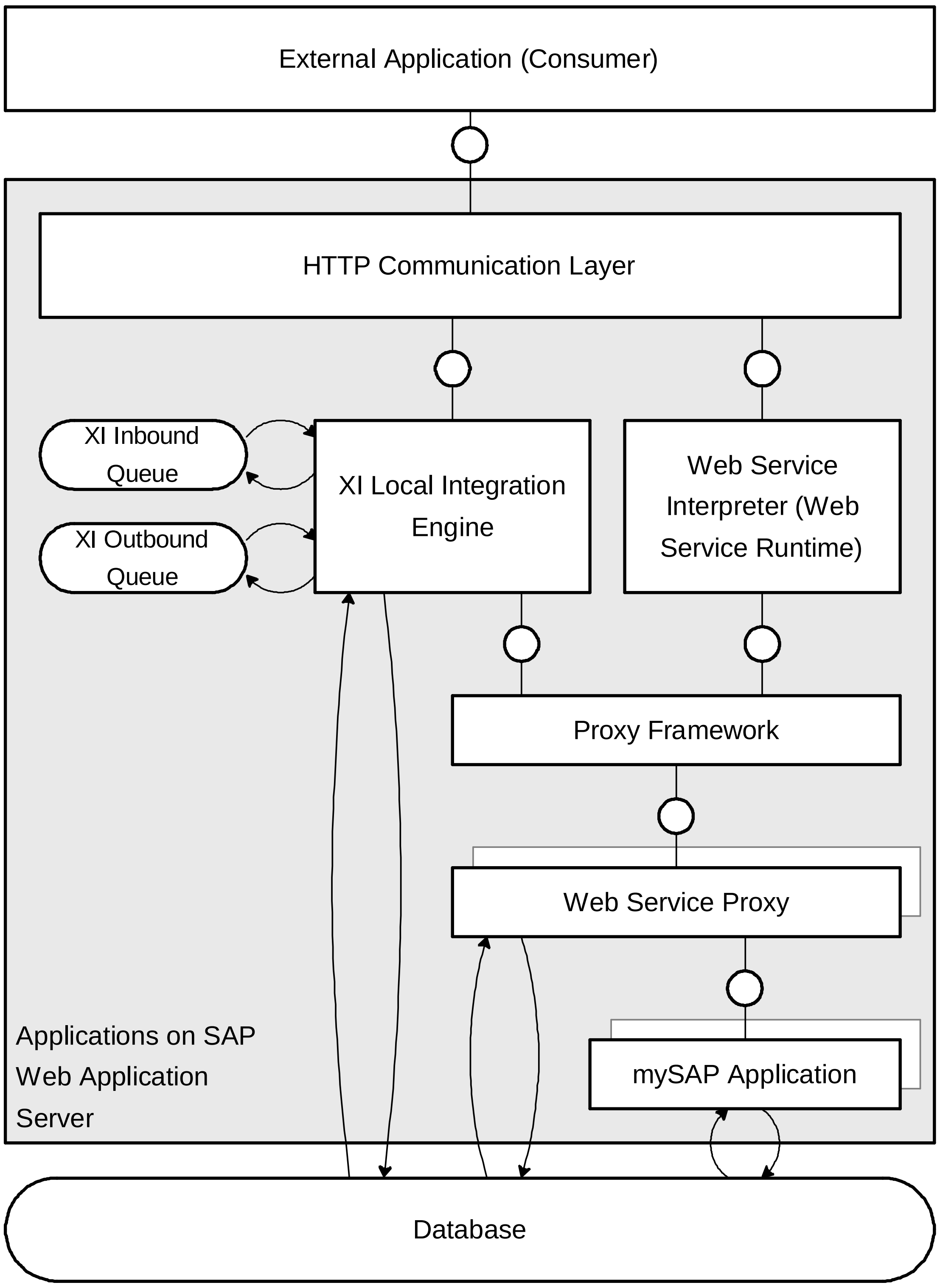
To employ services as idempotent services, you must use the MessageHeader element and its subelement UUID in the inbound and outbound message types. The UUID values must be globally recognizable and satisfy the format that is specified, for example, in IETF RFC 4122 (http://www.ietf.org/rfc/rfc4122.txt). If you do not use both elements when you call the services, the services do not behave in an idempotent manner.

Commit Procedures

The figure below shows all the relevant agents involved in the execution of a Web service call. As can be seen from the diagram, in addition to the mySAP Business Suite applications, two different agents have modifying access to the database: the *XI Local Integration Engine* and the *service proxy*. While the mySAP Business Suite applications need modifying access in order to write their data modifications to the database, these other two agents are responsible for committing the data modifications to the database in different scenarios.

- Asynchronous scenario: Commit by XI Only
- Synchronous scenario: No Commit by Framework

Commit Procedures


External Application (Consumer)
HTTP Communication Layer
XI Inbound Queue
XI Outbound Queue
XI Local Integration Engine
Web Service Interpreter (Web Service Runtime)
Proxy Framework
Web Service Proxy
mySAP Application
Applications on SAP Web Application Server
Database

Commit by XI Only

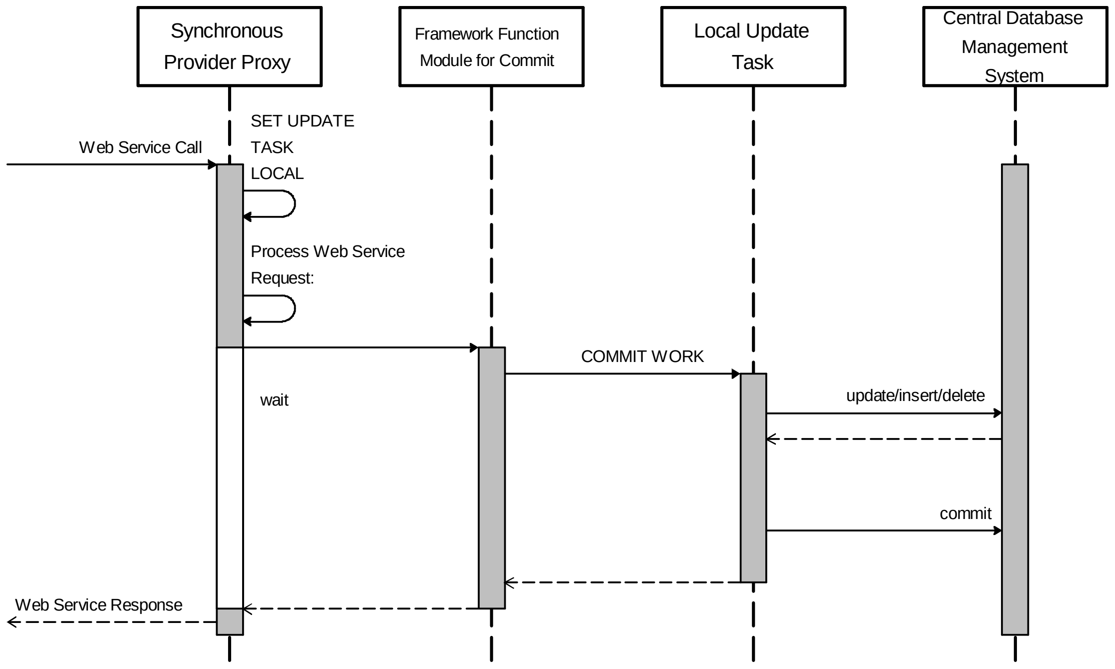
In the case of asynchronous service requests via SAP XI, the XI Local Integration Engine takes care of the transaction handling. It always opens a transaction before passing control to the proxy framework, and triggers a commit when the request processing is completed by the service proxy and control is handed back by the proxy framework.

Therefore, asynchronous service proxies must not perform a commit or call any function module which implicitly does so.

No Commit by Framework

For synchronous and point-to-point service calls, the commit is not carried out by any framework agent. Even if the service is called synchronously via SAP XI, the XI Local Integration Engine assumes that a synchronous call has the semantics of a query-response call, meaning that there is no modifying database access. Hence it will not trigger a commit. Therefore, in the case of synchronous services, the commit has to be implemented explicitly in the service proxy. The following figure shows the general process.

Regular Process Flow in Service Proxy Implementation

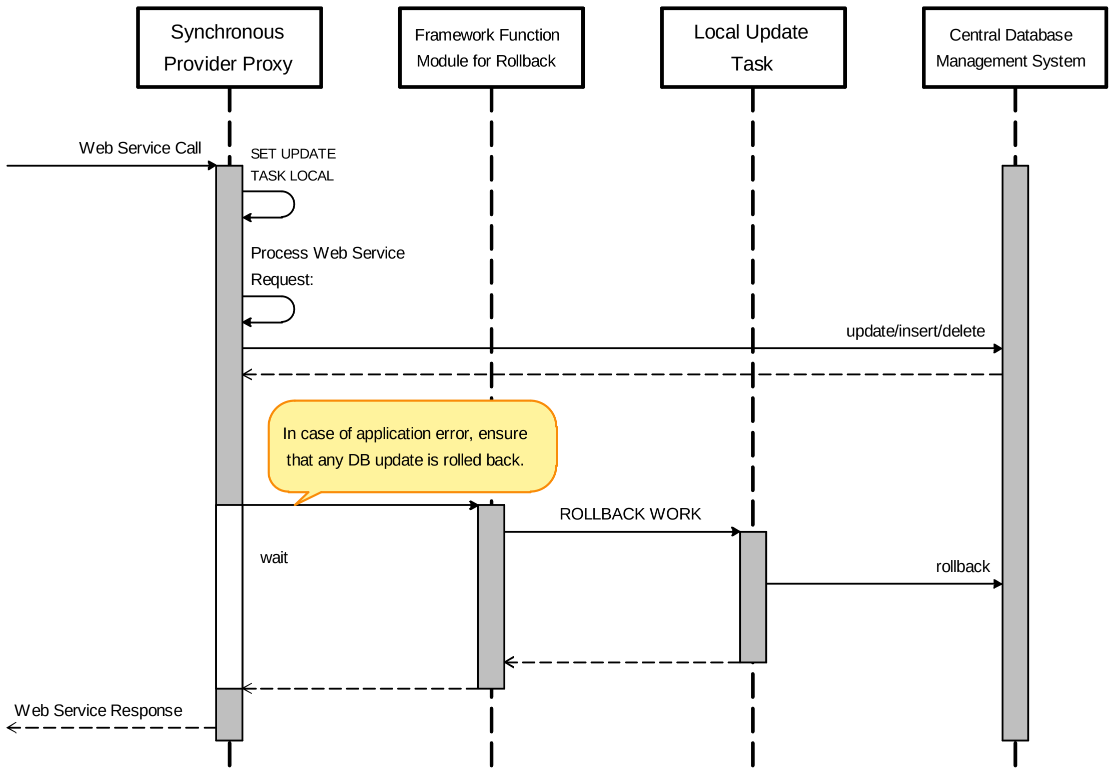

Synchronous Provider Proxy
Framework Function Module for Commit
Local Update Task
Central Database Management System
Web Service Call
SET UPDATE TASK LOCAL
Process Web Service Request:
COMMIT WORK
wait
update/insert/delete
commit
Web Service Response

After the actual processing of the Web service request, the service proxy triggers the commit. However, it does not trigger the commit process directly. The invocation of the commit process is encapsulated in a framework function module. This provides the flexibility to alter the transaction handling later on without having to change each service proxy implementation. For example, it leaves room for providing general transaction handling in the proxy framework at a later stage.

At the beginning of processing, the service proxy calls the ABAP statement SET UPDATE TASK LOCAL. This causes the commit to be carried out in the so-called local update task, synchronously within the same operating system process (called “work process” in SAP Web AS). The advantage of a synchronous commit is that the commit is guaranteed to be completed before the response is returned. Hence, information about the success of the commit can be included in the response message and it is ensured that in a subsequent reading call up-to-date data is read. Additionally, no separate operating system process (called “update work process”) is needed, and thus overhead is reduced.

In some cases, direct database access during the actual processing of the Web service request cannot be avoided. For example, if a new object is created, it might need a unique identifier, so that it can be put in relation to existing objects. When such a unique identifier is to be drawn from a continuously increasing number schema without gaps, it must persist in the database immediately to make sure that nobody else assigns the same unique identifier to another newly created object of the same type. This is depicted in the following figure by the direct call from the synchronous server proxy on the left to the database management system on the right.

Regular Process Flow in a Service Proxy Implementation


Synchronous Provider Proxy
Framework Function Module for Rollback
Local Update Task
Central Database Management System
Web Service Call
SET UPDATE TASK LOCAL
Process Web Service Request:
update/insert/delete
In case of application error, ensure that any DB update is rolled back.
ROLLBACK WORK
wait
rollback
Web Service Response

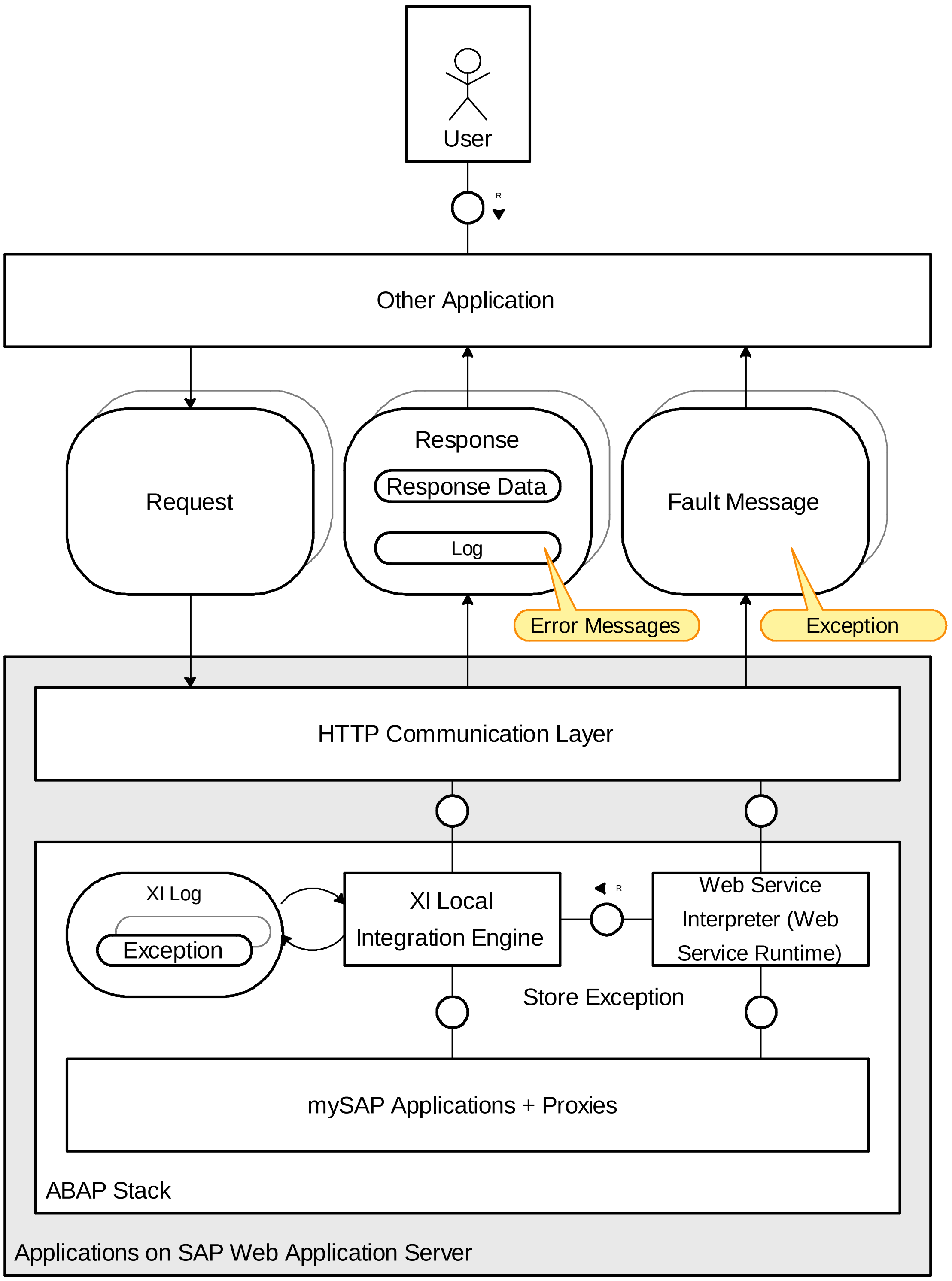
If an error occurs during subsequent processing, all database modifications of the current Web service execution need to be rolled back. For this purpose, the service proxy has to call a framework function module in case of errors. This function module takes care of the database rollback. Again, the rollback is encapsulated in a separate function module to allow for alteration of the transaction handling in the future.

Exceptions and Errors

During the execution of a Web service provided by mySAP applications, different types of errors can occur. Depending on the nature of the error, the other application is notified via either:

- An exception
- An error message

Exceptions and error messages are returned in different ways. Error messages are contained in the normal response, whereas exceptions are sent instead of the normal response.



Exception

An exception is raised when the error cannot be solved by the user, for example code errors, or inconsistent database data. Exceptions are sent instead of the normal response, unlike errors which are contained in the normal response.

XI Log

Exceptions are not only sent to the other application in the synchronous case, but are also stored in the XI log. This is especially important for invoking Web services asynchronously, as no response is sent in this case. This log is managed by the XI Local Integration Engine (see Web Service Invocation via XI Message Broker). The stored exceptions can be accessed and viewed at anytime by using the XI monitoring tools.

Fault Message

In the definition of a service, it is possible to specify fault messages which are sent back instead of the normal response message. Such a fault message is used to return an exception. A standardized fault message, called StandardMessageFault, is used for all Web services provided in the enterprise services packages. Again, in the asynchronous case the fault message is not directly sent to the calling application but is stored in the XI Log for later retrieval.

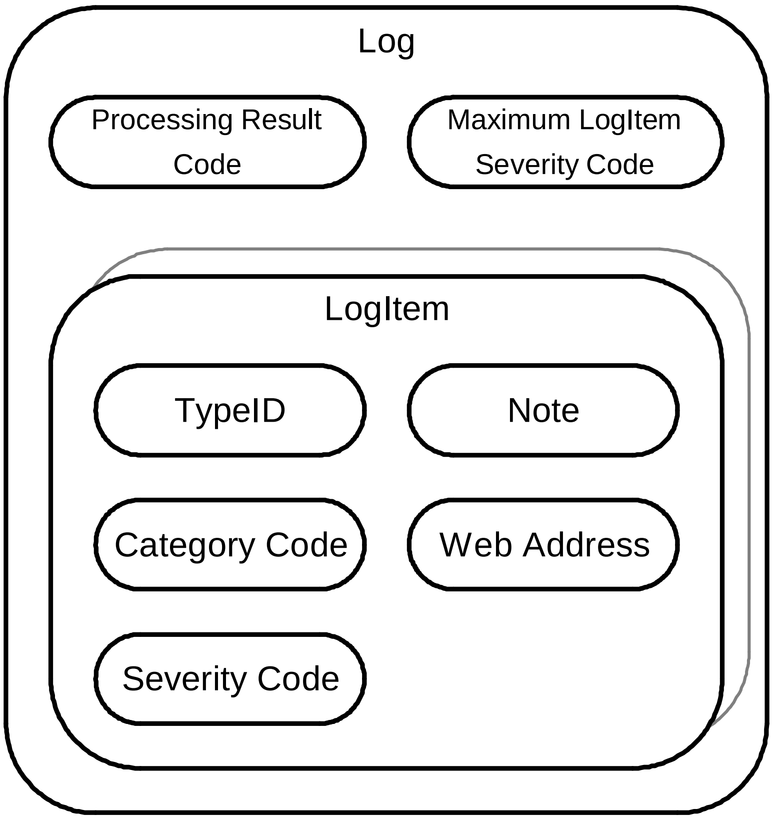
Error Message

Error messages are returned for problems that can be solved by the user, such as invalid input of data, authorization error, or locking error. Errors are contained in the normal response, unlike exceptions which are sent instead of the normal response.

Log Data Type

Including error messages in the normal response message is possible due to a convention. All return messages contain a specific data field, called log. This field is typed with a complex data type that is shown in the figure below.

Log Data Type



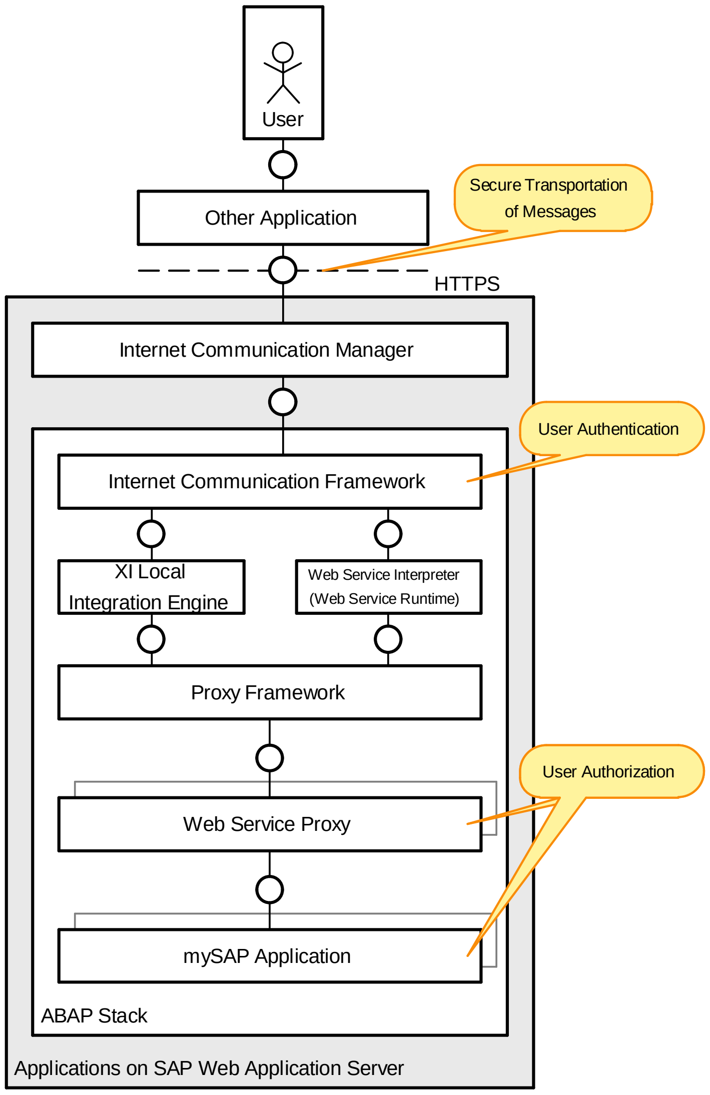
The processing result code specifies whether the Web service execution was successful, partially successful or failed. Depending on the code, the receiving other application can read and interpret the contained information and error messages. Each error message is transmitted as a single log item of the log. A log item can be assigned to an application specific type and category. The note of the log item contains the description of the error message. Any additional information about the error which is available on the Web is specified via the given Web address. Each log item is assigned to a severity level, which can be information, warning, error and abort. The highest level of all log items contained in a log is also stored in the log itself.

Security

The security of a Web service in mySAP Business Suite is accomplished in three areas:

- secure transportation of Web services messages
- authentication of Web service consumers in SAP Web AS
- authorization of Web service consumers within the mySAP Business Suite

Security in mySAP Web Services


User
Other Application
Secure Transportation of Messages
HTTPS
Internet Communication Manager
User Authentication
Internet Communication Framework
XI Local Integration Engine
Web Service Interpreter (Web Service Runtime)
Proxy Framework
User Authorization
Web Service Proxy
mySAP Application
ABAP Stack
Applications on SAP Web Application Server

More Information

As the subjects covered in these areas are not particular for Web services, the following sections only give an overview. For more information on security at SAP, see the SAP Service Marketplace at service.sap.com/security.

Providing Secure Transportation

This process is used to ensure secure transportation of Web service messages.

Process

To ensure secure transportation of Web service messages, a Secure Sockets Layer (SSL) connection for HTTP messages (HTTPS) can be used. This is the recommended option for secure message transportation in SAP NetWeaver 2004s. SAP NetWeaver 2004s partially supports the OASIS Web Services Security standard (XML Signature and authentication using UsernameToken). Future releases will fully support the OASIS Web Services Security standards, including XML-Signature and XML Encryption for securing the message content and Security Assertion Markup Language (SAML) 1.1 for single sign-on.

Providing Authentication in SAP Web AS

This process is used to authenticate Web service consumers in SAP Web AS.

Process

Consumers of ABAP Web services are authorized by the Internet Communication Framework. As the ICF only interprets HTTP and not SOAP, the information required for authentication must be contained in the HTTP part of a message.

SAP NetWeaver 2004s supports three different options for authentication of services:

- *Username and Password:* Service request messages contain the user name and password. ICF checks whether the user may access the system and the given password is correct.
- *SSL and Client Certificate:* This authentication option is based on a certificate that is used to prove the other application's (client) identity on the SSL connection from the other application to the SAP Web AS. The certificate contains a description of the identity of the other application and a public key that can be used by SAP Web AS to encrypt messages. With this key, SAP Web AS can establish the SSL connection with the other application and authenticate the user.
- *SAP Logon Ticket:* This authentication option is used for user authentication between different SAP Web AS instances. The different applications of the mySAP Business Suite may run on different instances of SAP Web AS. A user that is authorized in one instance of SAP Web AS does not have to authenticate itself again in another instance. Therefore, a ticket containing authentication information for Single Sign-On, called mySAP SSO2, is contained in the HTTP header of messages that are exchanged between SAP Web AS instances.

If the authentication of the Web service consumer is successful, the request is executed under this user's identity. This identity is used for authorization checks in subsequent calls to application functionality (see Providing Authorization within the mySAP Business Suite).

Public Web Services

It is also possible to define default users for Web services. In this case, the other application does not include any user data in the Web service message. The ICF assigns a predefined user to the incoming requests. Web services with default users are also called Public Web services, as everybody can access them without restriction.

Note

In future releases of SAP NetWeaver, other authentication options, such as user name and password in a SOAP message, XML-Signature and SAML will be supported. Some of these authentication options can only be provided by the Web service interpreter, as ICF cannot read the SOAP content of HTTP messages. For this reason, a standard user is assigned by the ICF during the normal authentication process. The Web service interpreter substitutes the real user for this standard user if the authentication is successful.

End of the note.

Providing Authorization within the mySAP Business Suite

This process is used to authorize Web service consumers within the mySAP Business Suite.

Process

Accessing mySAP functionality via Web services follows the standard mySAP authorization concept. This concept is based on authorizations for specific authorization objects. If a user does not have an authorization for an authorization object, which is checked during the execution of a Web service, the execution is terminated, and an error message is returned (see Exceptions and Errors).

Web Service Invocation via XI Message Broker

The enterprise services packages for the mySAP Business Suite on SAP NetWeaver 2004s focus on providing synchronous Web services. However, Web services provided or called by an mySAP application need not be invoked synchronously. The underlying technical infrastructure of SAP Web AS also supports asynchronous service invocations. However, in order to use them reasonably, a networking infrastructure that provides reliable messaging is required.

Features

- SAP Exchange Infrastructure (SAP XI)
- XI Protocol
- XI Local Integration Engine

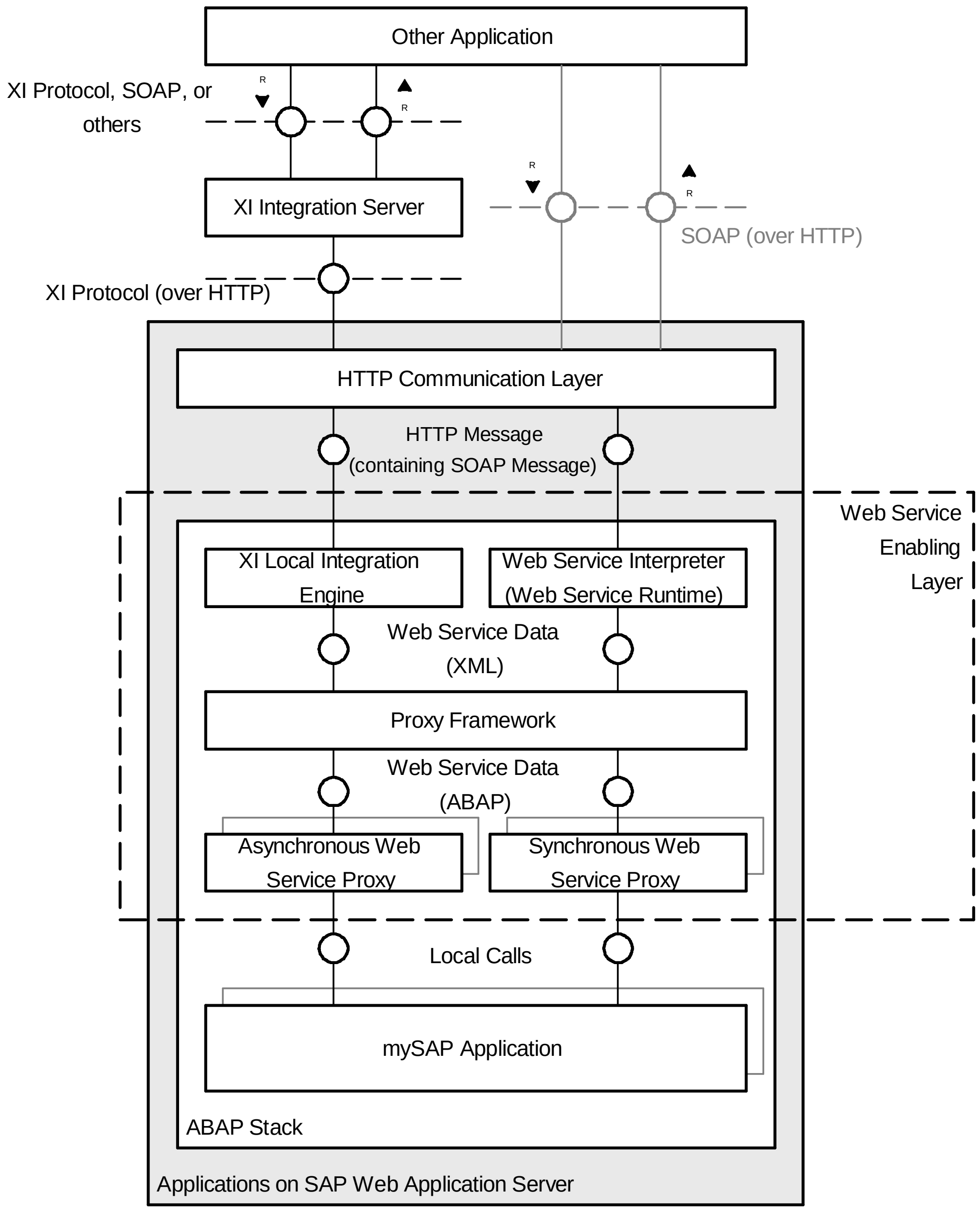
SAP Exchange Infrastructure

SAP Exchange Infrastructure (SAP XI) is an integration technology and platform. XI uses by default an extended SOAP protocol for connecting applications. In addition, other protocols, such as JMS, JDBC, RFC and IDOC, can be used with adapters. XI provides communication services such as message routing and mapping of messages between different formats and contains a component for cross-component business process management (ccBPM). This component makes it possible to model and execute business processes on XI.

SAP XI provides, among other things, reliable messaging for asynchronous communication. Therefore, the mySAP application does not implement such functionality on its own for the asynchronous invocation of Web services but requires SAP XI-based communication in this scenario.

The figure below shows the XI scenario for invoking Web services by refining the architecture introduced in the previous sections. The different types of proxies described in the section Proxy Type are not shown (for clarity reasons).

XI Based Web Service Invocation


Other Application
XI Protocol, SOAP, or others
R
R
XI Integration Server
R
R
SOAP (over HTTP)
XI Protocol (over HTTP)
HTTP Communication Layer
HTTP Message (containing SOAP Message)
Web Service Enabling Layer
XI Local Integration Engine
Web Service Interpreter (Web Service Runtime)
Web Service Data (XML)
Proxy Framework
Web Service Data (ABAP)
Asynchronous Web Service Proxy
Synchronous Web Service Proxy
Local Calls
mySAP Application
ABAP Stack
Applications on SAP Web Application Server

More Information

For more information about SAP XI, see the SAP help portal at help.sap.com *SAP NetWeaver.*

XI Protocol

The XI Integration Server understands all common network communication protocols, and application protocols, such as HTTP and SOAP. In addition, it offers the specific XI protocol on the application level. This protocol is based on SOAP and enhances it with information required to provide network infrastructure services, such as message routing, reliable messaging, and message monitoring. These services are not available in point-to-point scenarios that rely on the standard SOAP protocol. By default, the XI Integration Server is accessed via the XI protocol. However, other protocols can be utilized by using adapters.

Communication between the XI Integration Server and SAP Web AS is always carried out using the XI protocol. As this protocol is an enhanced SOAP protocol and based on HTTP, the HTTP Communication Layer can also handle XI protocol messages without modification.

XI Local Integration Engine

The XI Local Integration Engine is the receiver of the XI messages in the Web Service Enabling Layer, instead of the Web service interpreter.

This local counterpart to the XI integration server speaks to the XI protocol and extracts the Web service data from the message. As the XI protocol also provides services such as reliable messaging, the XI Local Integration Engine does not only transform and forward the incoming message, but may also trigger the XI integration server to retransmit messages that have been lost.

Once the Web service data has been extracted from the XI message, the rest of the Web service execution is identical to the execution in the point-to-point scenario (see Runtime Architecture Overview).

Note

By using standardized extensions to the SOAP protocol, it is possible that some of the mentioned services, such as reliable messaging, can be provided in point-to-point scenarios. However, these standardized extensions are not supported in SAP NetWeaver 2004s, but in later releases only. The most common standardized extensions are Web Service Reliable Messaging, Web Service Addressing and Web Service Security.

End of the note.

mySAP Service Provisioning: Design Time

Services in the enterprise services packages are created in two steps during design time.

Process

First, implementation independent service interfaces are defined. These service interfaces are aligned through a common object model representing the business content of the mySAP Business Suite.

See Business Object and Service Modeling

In order to realize the defined services as Web services working on mySAP applications on the ABAP stack, corresponding ABAP entities have to be created and implemented.

See Proxy Generation in the ABAP Stack

Business Object and Service Modeling

This process is the first step in creating services in the enterprise services packages.

Process

This process involves the definition of implementation-independent service interfaces by creating a model of the business functionality to be offered, determining the functions to be realized, and deriving the appropriate service interface definitions.

- Business Object Modeling
- Service Interface Modeling

Governance Process

Since a sound model of the business content is crucial for SAP's success in service provisioning, SAP has set up a governance process for the modeling activities. The governance process ensures that domain experts from all concerned divisions of SAP have reviewed the model and that the model is harmonized across SAP as far as possible.

ESR

The approved service interface definitions are stored in the Enterprise Service Repository (ESR). The ESR is SAP's design time repository for reusable type definitions.

ARIS

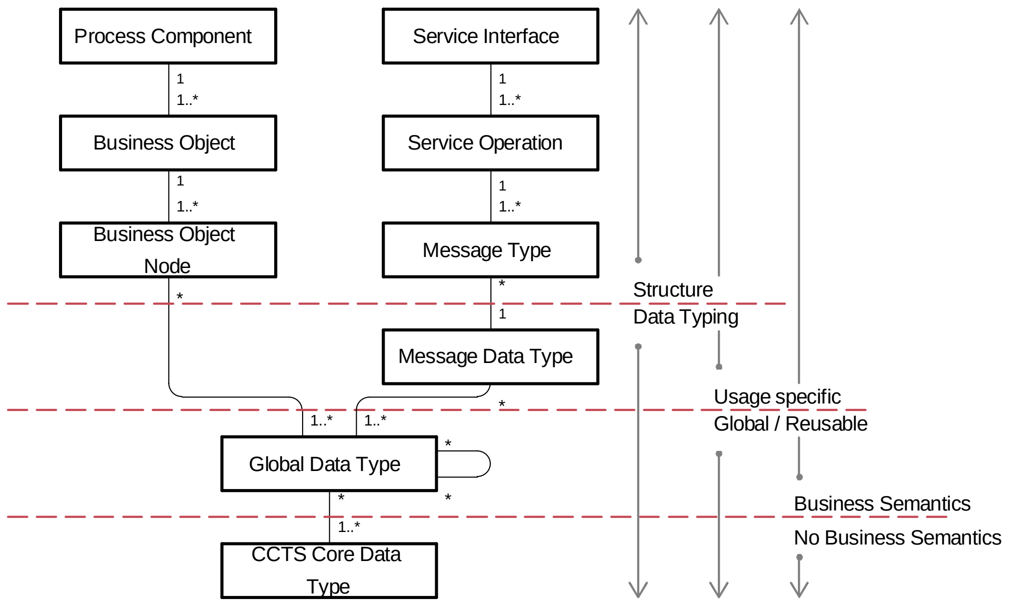
High-level models are kept and maintained in the modeling tool ARIS and can be viewed in the Enterprise Services Workplace in the SAP Developer Network (SDN) (at sdn.sap.com *sdn.sap.com* *ES Workplace*).

Business Object Modeling

This process is used to create a model of the business functionality to be offered, determining the functions to be offered.

The entities in the business model of SAP application functionality are shown in the figure below.

Entities of Service Interfaces and Business Object Models



Process

Process components and business objects are modeled in the ARIS modeling tool and can thus be viewed in the Enterprise Services Workplace. The internal structure of the business objects is only modeled SAP-internally and not released to the customers.

1. Process Component
2. Business Object

Process Component

A process component comprises all functionality offered by a service-oriented SAP application that realizes a certain business process, where a business process is regarded to be a sequence of activities transforming a defined business input into a defined business outcome.

Integration

A set of business objects belongs to each process component.

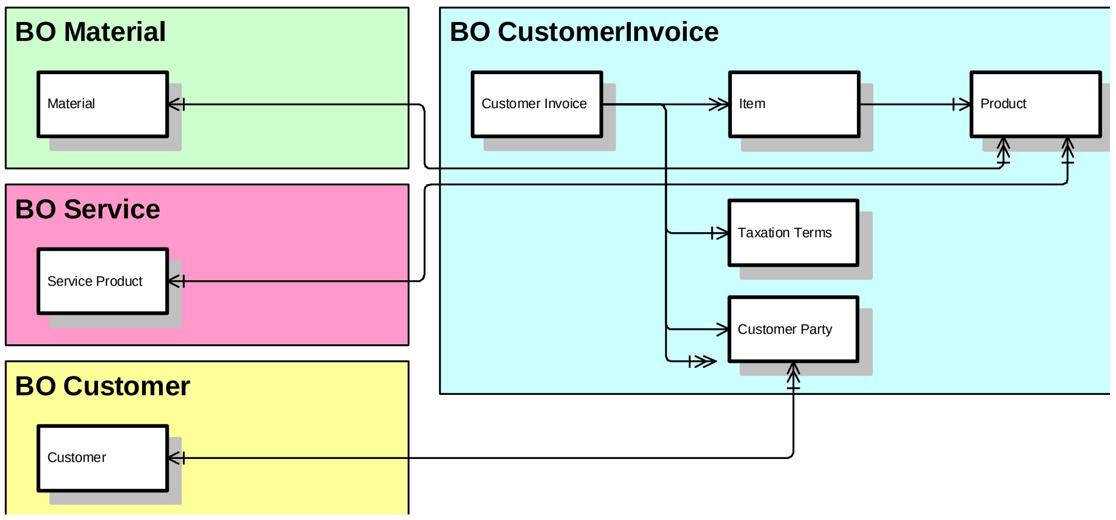
Example

An example for a process component is “customer invoice processing”.

Business Object

Business objects are entities from the business world that are relevant for a business process. A business object may be of diverse nature. Examples for business objects are documents, such as “customer invoice” or “employment” contract, legal entities, such as “customer” or “bank”, or physical objects such as “material” or “inventory” item.

Example of a Business Object Model



Structure

Business Object Node

Each business object is modeled as a hierarchical structure of nodes. As an example, the figure above shows an extract of the business object model of the customer invoice.

Nodes may have different cardinalities depending on their business semantics. For some nodes an arbitrary number of node instances is possible, like for line items in a customer invoice. For other nodes only a limited number of instances may exist. For example, each line item of a customer invoice represents only a single product, which is to be invoiced. In addition to the internal structure of a business object, the associations between related business objects are modeled as well in the business object model. For example, the customer invoice depends on a customer and a material or a service product.

Global Data Type (GDT)

Each business object node is made up of data fields, which are typed with so-called global data types (GDTs). GDTs are allowed and intended to be reused in different business object nodes. The idea is that for each basic business fact a GDT is defined and used wherever the business fact has to be represented in a data field.

GDTs may be either unstructured (for example, ProjectName and TaxableIndicator) or structured, that is a record of two or more other GDTs (for example, DeliveryTerms and Address).

CCTS

Unstructured GDTs are either qualified versions of other unstructured GDTs or represent a core data type as defined in the Core Components Technical Specification (CCTS) by UN/CEFACT (http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf) .

Core component types are basic data type definitions on a more technical level, which have no business semantics on their own. Examples for core data types are identifier (a character string uniquely identifying a certain object within a certain context), indicator (a Boolean value) or amount (an amount of money together with the currency). By adhering to a public standard for elementary data types, it is easier to exchange data with third party applications.

Example

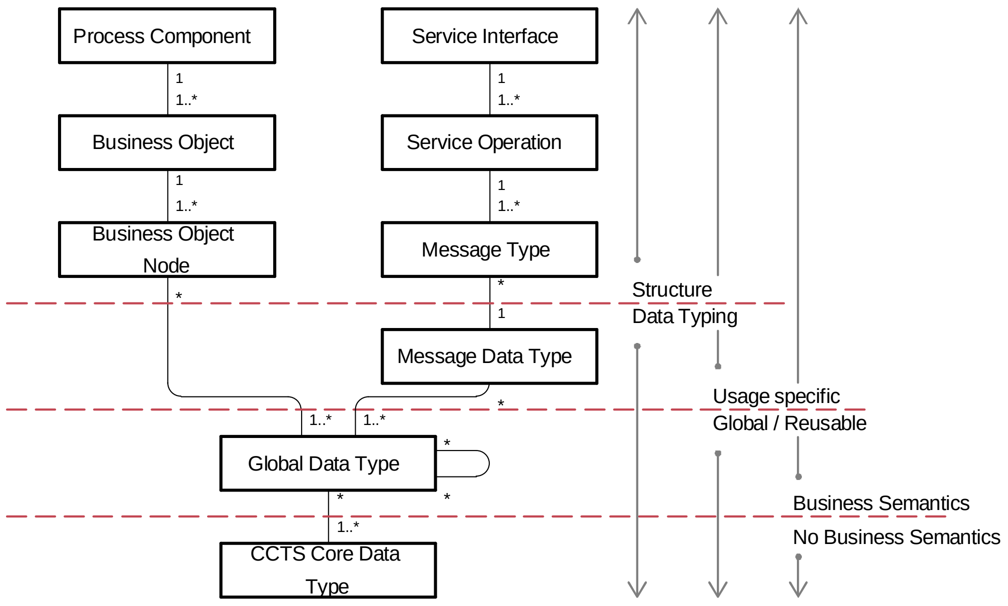
Examples for business objects are documents, such as “customer invoice” or “employment” contract, legal entities, such as “customer” or “bank”, or physical objects such as “material” or “inventory” item.

Service Interface Modeling

This process is used to derive the appropriate service interface definitions.

Service interface definitions for enterprise services have a uniform hierarchical structure, as shown below.

Entities of Service Interfaces and Business Object Models



Service Operation / Message Interface

For each operation, up to two parameters may be required, an input parameter and an output parameter. The input parameter has to be provided when calling the service. The output parameter is determined during the service execution and returned to the caller.

In addition to these two parameters per operation, an additional universal fault parameter exists. This fault parameter is sent back to the consumer in case of exceptions instead of the normal output parameter (see Exceptions and Errors).

Service Operations are also called Message Interfaces.

Message Type

The parameters are also called messages, since they make up the payload of the SOAP messages transferred between the consumer and the provider. For each service, the message required as input parameter and/or the message provided as output parameter must have a certain message type. The universal fault parameter has its own separate message type. Except for the fault message type, message types must not be reused between service operations, but are unique to one service operation.

Message Data Type

The message types are typed with a message data type, which is a record of data fields typed with GDTs, analogous to the data fields of business object nodes. Message data types unlike message types may be reused.

Process

Leading Business Object

All services are grouped around the business objects. Each service operation is associated with the business object that is central for the service operation's business purpose. For example the service operation customer invoice creation request is associated with the business object customer invoice. There may be multiple business objects relevant for the execution of a service operation. In the customer invoice example, the customer and the products to be invoiced are relevant as well. The service operation will be assigned to the central business object, which has to be identified among all relevant business objects. This central business object is also called *leading business object* for the service operation.

Message data types for a service operation are always derived from the leading business object as described in the following paragraphs. This ensures that service operations that are associated with the same central business object always use compatible message data types.

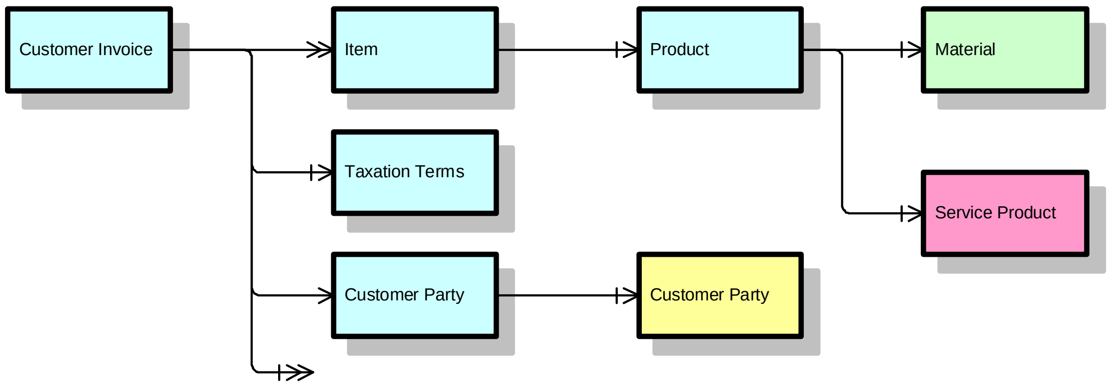
A message data type consists of up to three parts:

- a generic message header
- a so-called business document object
- for output messages: a generic log structure, which is used to return administrative information about the service execution.

Business Document Object

The business document object contains the specific business content of the message, that is the business facts relevant for the service execution or the business facts that show the result of the service execution. The business document object is the part of the message data type that is derived from the business object model. The business document object is a trimmed extract from the business object model containing all relevant parts of the leading business object of the service interface and all relevant parts of related business objects. The following figure shows the business document object corresponding to the customer invoice example above.

Example of a Business Document Object



Since the business document object is always a part of the message data type, it has to be structured hierarchically. The root node of the leading business object becomes the root node of the business document object. Below the root node, all relevant parts of the leading business object are included in the business document object according to the business object model. The required parts of associated business objects are linked below those nodes of the leading business object, with which they are associated in the business object model. The extent to which associated business objects are included in the business document object depends on the semantics of the service operation. In some cases an identifier of the object is sufficient. In other cases one or more nodes or even the whole business object may be required.

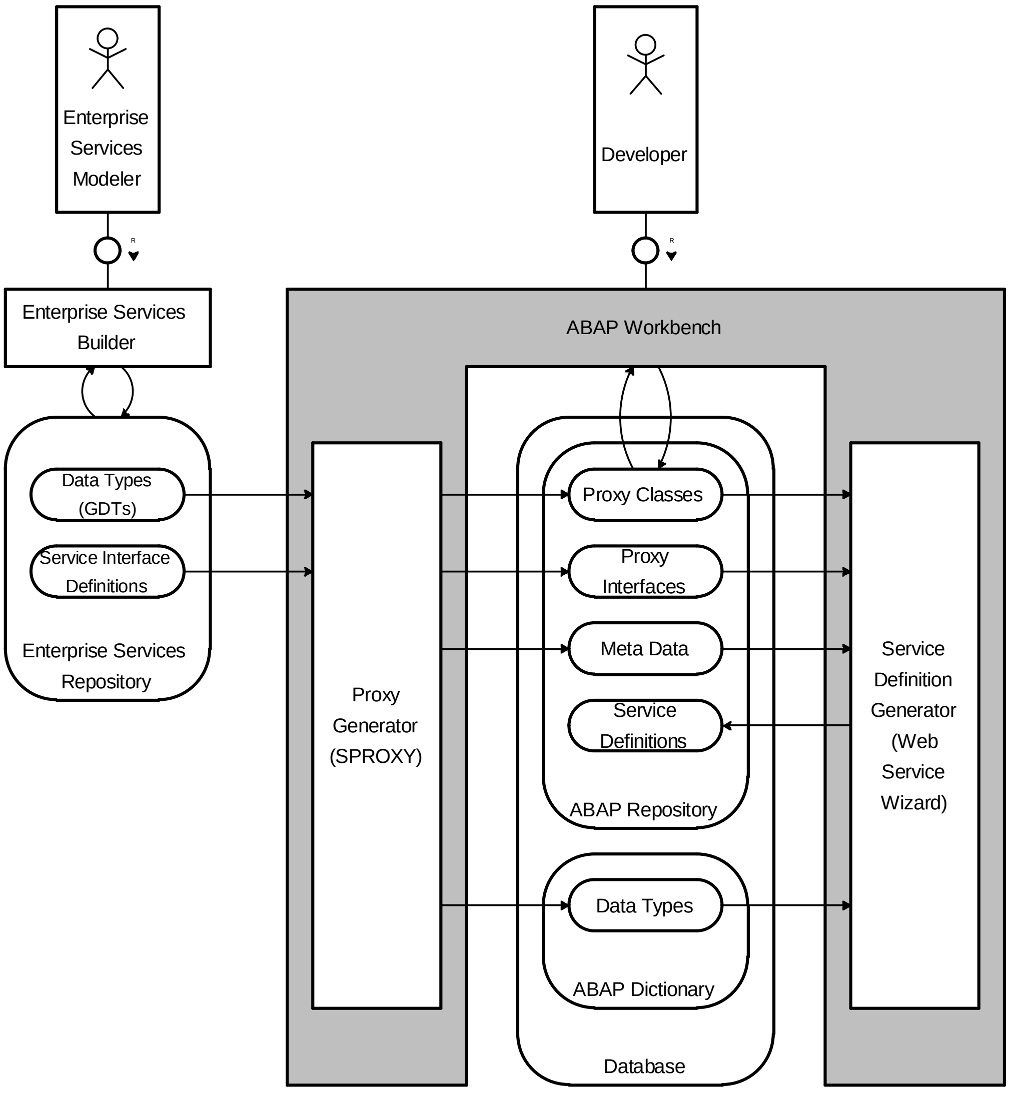
Proxy Generation in the ABAP Stack

A service interface definition in ESR is an abstract definition that does not depend on a specific implementation. In order to use the abstract definitions in the ABAP stack, corresponding ABAP entities have to be created and implemented in SAP Web AS.

ABAP Entities and Meta Data

In the ESR, the structure of a Web service is defined by specifying a service interface and the data types that are used in this interface (see Business Object and Service Modeling). Corresponding ABAP entities for the service interfaces and data types defined in ESR are ABAP interfaces, ABAP classes and ABAP data types. In addition meta data of the ABAP entities are required in the ABAP stack in order to execute a Web service. The meta data contains information about the entities that have been created based on ESR content. This information specifies for example how to map data types from the XML representation in ESR to the ABAP representation.

Design Time Architecture



Process

1. *Enterprise Services Builder*

   The figure above shows the systems involved in designing service interfaces in ESR and creating ABAP entities in the ABAP Stack on SAP Web AS for providing mySAP Web services. Service interface definitions and data types are created in ESR with the enterprise services builder.

2. *Proxy Generator*

   Based on the ESR content, the proxy generator creates all necessary ABAP entities for a service interface. It creates, on the one hand, the required ABAP data types in the ABAP dictionary, on the other hand, the ABAP interface and the ABAP class for the proxy which implements the service interface.

3. *Automatic and Manual Proxy Implementation*

   The implementation of a consumer proxy can be generated automatically.

   A provider proxy which relies on existing functionality must be implemented by a developer, as the logic which calls the existing functionality can not be created automatically. Therefore, the proxy generator only creates an empty ABAP class for provider proxies. The methods of this class have to be implemented by a developer. Each method corresponds to an operation of the Web service.

4. *Service Definition Generator*

   Once the proxy classes have been implemented, a service definition has to be created with the Web service definition generator, before a Web service can be released (see Releasing a Web Service). A service definition contains the design time properties of a Web service. The properties specify preferred security settings which should be used when the Web service is released during configuration. Design time properties, which are created with the service definition generator, are stored in a service definition entity.

mySAP Service Provisioning: Configuration

This process is used to configure services created by SAP. Once the services have been created by SAP, they are delivered with the enterprise services packages to the customer. The customer needs to configure the services according to his or her needs and the available IT infrastructure.

Business Configuration

The business configuration of the service-enabled mySAP Business Suite does not differ from that of the conventional mySAP Business Suite. Details of the business configuration of the mySAP Business Suite can be found in its Implementation Guide (IMG). The implementation guide is a tool for configuring the SAP system to meet customer requirements. It is accessed from the SAP menu of the SAP system by choosing *Tools* → *Customizing* → *IMG*.

SAP XI

For services that are to be called via SAP XI, the conventional XI configuration has to be carried out. For more information about SAP XI, see the SAP help portal at help.sap.com → *SAP NetWeaver*.

Process

- Releasing a Web Service
- Performing Web Service Administration
- Configuring Idempotent Services

Releasing a Web Service

This is the first step of the configuration process.

Note

You perform the configuration described in these steps using the WSCONFIG transaction. For more information, see the SAP Help Portal under help.sap.com *SAP NetWeaver* *Releasing for the SOAP Runtime* .

End of the note.

Procedure

1. Activate the required services

   You need to decide which of the services delivered by SAP should be active in your installation of the mySAP Business Suite. These services have to be activated explicitly. The activation process is also referred to as the release of the service.

2. Apply variants

   For each service delivered, a service definition is provided. The service definition consists of a service interface description and one or more variants. When a service is to be released, the next step is to decide which of the variants available is applied.

   See Service Definition and Service Variant

3. Specify the following additional information:

   - *Name and release text:* Name and textual description of the service that will be published to the consumers.
   - *Virtual host:* The name of the SAP system which acts as the service provider. The virtual host implicitly determines the transfer protocol.
   - *URL:* The relative URL of the service in relation to the virtual host.
   - *Security profiles:* Security settings specifying the authorizations required for the execution of the Web service.

4. Generate a runtime configuration

   The service configuration utility takes the service interface description, the chosen variant and the additional information to generate a WSDL description and a runtime configuration. The WSDL description can be published to the service consumers, while the runtime configuration is needed by the Web service interpreter and the proxy framework to process calls to the service.

Service Definition and Service Variant

For each service delivered, a service definition is provided. The service definition consists of a service interface description and one or more variants. A variant describes the features of the service. Features in this context are properties of the service on a non-technical level, which restrict the choice of host and protocol in the configuration process. For example, if the variant specifies that the service needs authentication by certificates, it is not possible to use HTTP as a transfer protocol. HTTPS should be used instead.

Features

There are three feature types for services:

- *Unique message ID:* A unique message ID should be assigned to each message.
- *Authentication level*: Requirements for the authentication mechanism. Possible values are: "no authentication required", "password authentication required" and "certificate authentication required".
- *Transport guarantee level:* Requirements for the transport mechanism. It specifies whether integrity and/or confidentiality have to be ensured. Integrity is the guarantee that transferred messages arrive unmodified at the receiver. Confidentiality is the guarantee that only the receiver can access the unencrypted content of a transferred message.

Activities

When a service is to be released, the first step is to decide which of the variants available is applied.

Performing Web Service Administration

The administration of Web services is the part of the configuration which takes place when the Web services are released.

1. Make the following two administration settings for an SAP Web service
   - *Trace and Log:* The extent to which information about the use of a service is recorded. It is possible, for example, to record information about calls to the service received, errors that occur during service execution, and performance measures.
   - *UDDI publication:* The WSDL description of released Web services can be published in UDDI repositories.
2. Test the Web service

   SAP's Web service administration utility provides a test Web site, which allows invoking the Web service with test parameters. This can be used to check the configuration settings applied.

Note

You perform the administration described below using the WSADMIN transaction. For more information, see the SAP Help Portal under help.sap.com *SAP NetWeaver* *Administration for the SOAP Runtime.*

End of the note.

Configuring Idempotent Services

You use this method to set how long the system stores saved response messages or saved information about responses that have been sent. An initialization process takes you through the steps the first time you make these settings.

The initialization process incorporates the following steps:

- The system creates several database objects that are necessary for saving response messages.
- The system schedules the following background jobs for the regular deletion of saved response messages and message IDs:
    - SAP_BC_IDP_WS_SWITCH_BD
    - SAP_BC_IDP_WS_SWITCH_BDID

To ensure, for performance reasons, that the responses or the information about sent responses are deleted as soon as possible after the desired period, the system uses table pairs:

- two tables per client for response messages
- two tables per client for information about response messages that have been sent

The two tables are filled alternately. When changing from the first table to the second table, the data in the first table is kept, but the data from the second table is deleted before the system adds new content. You decide how often the background jobs change tables using the procedure described here. This results in a minimum storage duration for the data.

Prerequisites

You must be using at least SAP NetWeaver 2004s SPS09.

Procedure

1. Start transaction WSIDPADMIN.
2. In the *Document* area, enter the minimum storage period for response messages. In addition, enter how often the tables must be changed.
3. In the *Document ID* area, enter the minimum storage period for information about response messages that have already been sent. In addition, enter how often the Document ID tables must be changed. We recommend you choose a longer time interval than you did for the response messages. Note that a repeated request message results in an error if the corresponding response message has already been deleted but the information about the sent response message is still available.
4. Execute the transaction by choosing *Schedule* or pressing F8.
5. Repeat the settings in every client in which you want to call Enterprise Services.

Example

You use a time period of two hours for changing the document tables. This means that response messages are stored for at least two hours but at most for just less than four hours.

You use a time period of one day for changing the document ID tables. This means that information about sent response messages is stored for a maximum of just less than two days.

Example: Service Operation “Create Employee Leave Request”

This section provides a walk-through of service provision using the example of the service operation “Create Employee Leave Request”. This service operation is associated with the business object “EmployeeLeaveRequest”, which belongs to the process component “Time and Labour Management”.

Process

1. Modeling of Process Components in ARIS
2. Business Object Modeling
3. Modeling of Service Interfaces in ESR
4. Proxy Implementation

Modeling of Process Components in ARIS

The first step in the process of service creation is the development of the process component model in the modeling tool ARIS.

Procedure

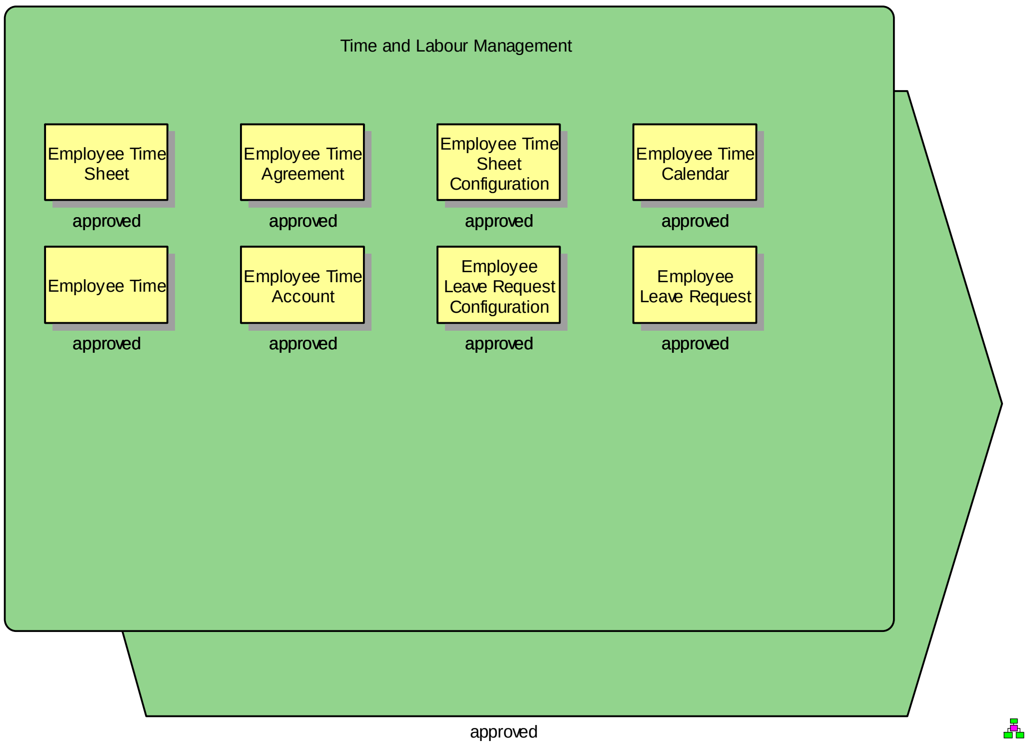
The first figure below shows the process component “Time and Labour Management” as an extract from the business object map, which is an overview of all process components and business objects.

Time and labour management supports the definition of employees' planned working time as well as the recording of the actual working times and absences and their evaluation.

Inside the process component, the contained business objects are shown. One of them is the business object EmployeeLeaveRequest.

An EmployeeLeaveRequest is the application form used by an employee to request leave. The request contains information about the desired leave besides request information such as submission date or the selected approver. The approver receives information about the desired leave and (depending on the type of desired leave and its configuration) has the possibility to approve or reject the request.

Process Component & Business Objects

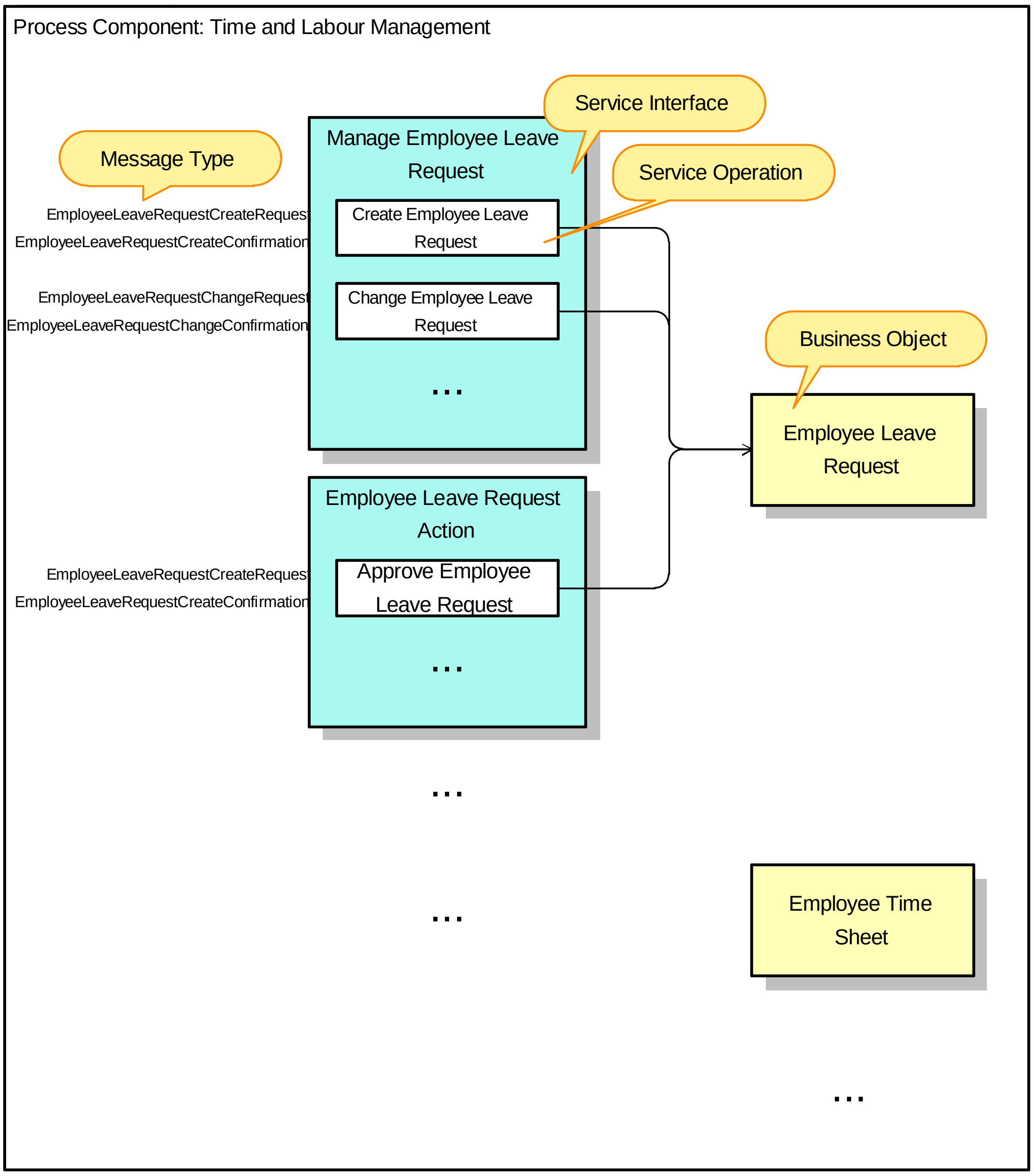


The next figure below shows an extract of the process component model of the process component “Time and Labour Management”. The process component model particularizes the business object map. For one process component, it shows all contained business objects together with the related service interfaces. For each service interface the operations and messages are shown.

For the business object EmployeeLeaveRequest, one of the service interfaces is the interface “Manage Employee Leave Request”, which contains all operations necessary to write, change and delete employee leave requests. Another service interface for the business object EmployeeLeaveRequest is “Employee Leave Request Action”, which contains operations for approving and denying employee leave requests.

One of the operations in the service interface “Manage Employee Leave Request” is the operation “Create Employee Leave Request”, which triggers the creation of a new employee leave request. The operation makes use of two messages, the inbound message EmployeeLeaveRequestCreateRequest and the outbound message EmployeeLeaveRequestCreateConfirmation.

Process Component Model


Process Component: Time and Labour Management
Message Type
Service Interface
Service Operation
Manage Employee Leave Request
Create Employee Leave Request
Change Employee Leave Request
EmployeeLeaveRequestCreateRequest
EmployeeLeaveRequestCreateConfirmation
EmployeeLeaveRequestChangeRequest
EmployeeLeaveRequestChangeConfirmation
Business Object
Employee Leave Request
Employee Leave Request Action
Approve Employee Leave Request
EmployeeLeaveRequestCreateRequest
EmployeeLeaveRequestCreateConfirmation
Employee Time Sheet

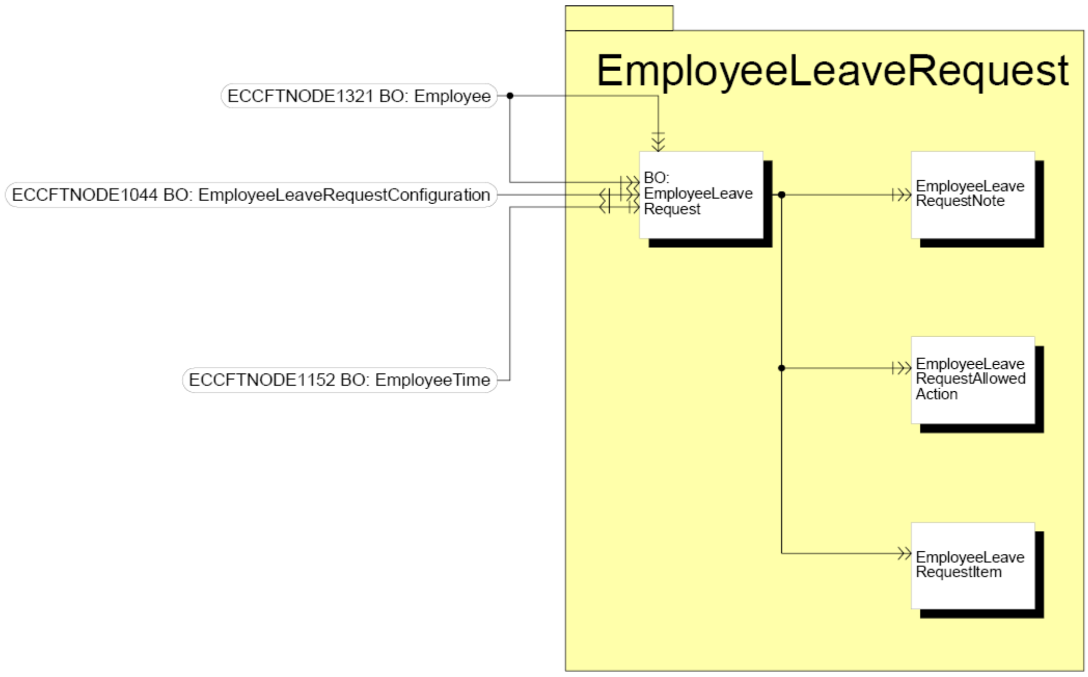
Business Object Modeling

The second step in the process of service creation is to define the inner structure of the business objects in a business object model.

Procedure

The figure below shows the business object model for the business object EmployeeLeaveRequest. Besides the root node, there are three sub nodes: Note, AllowedAction and Item.

Business Object Model


EmployeeLeaveRequest
ECCFTNODE1321 BO: Employee
ECCFTNODE1044 BO: EmployeeLeaveRequestConfiguration
ECCFTNODE1152 BO: EmployeeTime
BO: EmployeeLeave Request
EmployeeLeave RequestNote
EmployeeLeave RequestAllowed Action
EmployeeLeave RequestItem

A *note* is a free text together with information about its author and the date and time of creation.

The *allowed action* is a coded representation of the way in which the transmitted document can be processed, for example “delete”, “modify” or “approve”.

An *item* of an employee leave request is a document item describing the details of an employee’s requested leave, e.g. the kind of leave and the planned period.

The business object EmployeeLeaveRequest has four associations with other business objects: to EmployeeTime, to EmployeeLeaveRequestConfiguration and two associations to Employee.

The linked employee time entry is an existing one, which is the result or the subject of this employee leave request.

The linked employee leave request configuration provides information about the functional and UI arrangement of employee leave requests concerning the individual business needs of the company.

One of the linked employees is the requester. The other linked employees are further participants, for example the responsible approver of the request.

The data fields contained in the business object nodes are not discussed here individually, but can be seen from the message structure in the next section about service interface modeling.

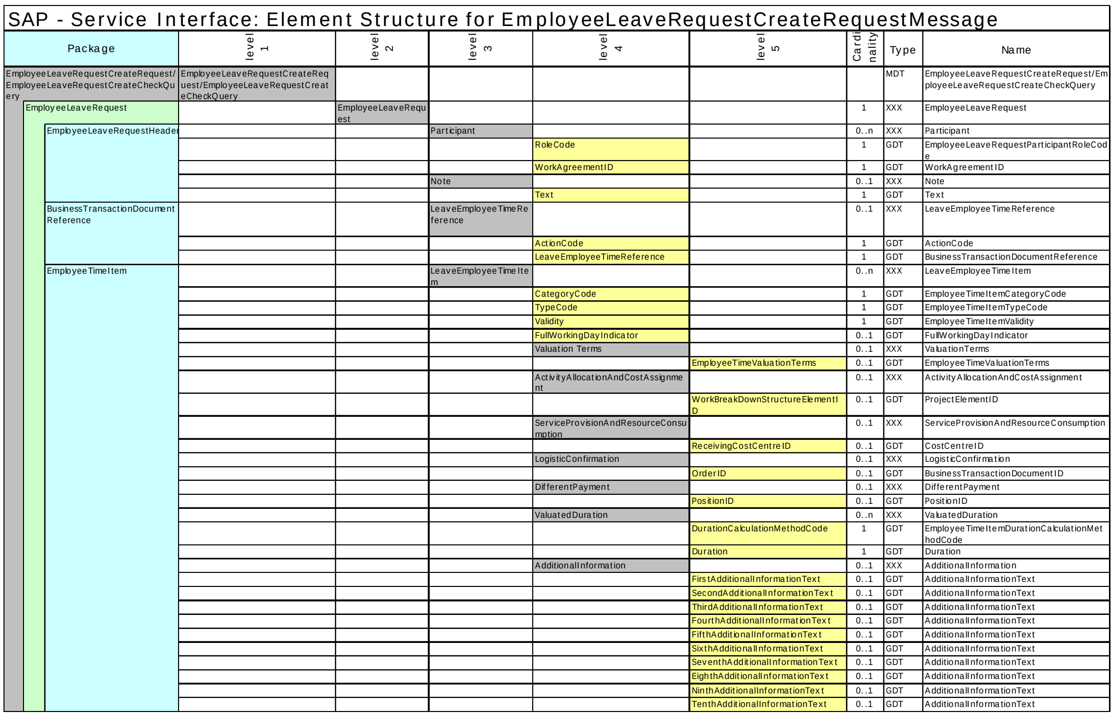
Modeling of Service Interface in ESR

The third step in the process of service creation is to derive the structure of the messages from the business object model and register the service operations together with the required messages in ESR.

Procedure

The figure below shows the message structure for the “Employee Leave Request Create Request” message as it is registered in the ESR.

Message Structure



SAP - Service Interface: Element Structure for EmployeeLeaveRequestCreateRequestMessage

Package	level 1	level 2	level 3	level 4	level 5	Cardinality	Type	Name
EmployeeLeaveRequestCreateRequest/EmployeeLeaveRequestCreateCheckQuery	EmployeeLeaveRequestCreateRequest/EmployeeLeaveRequestCreateCheckQuery						MDT	EmployeeLeaveRequestCreateRequest/EmployeeLeaveRequestCreateCheckQuery
EmployeeLeaveRequest		EmployeeLeaveRequest				1	XXX	EmployeeLeaveRequest
EmployeeLeaveRequestHeader			Participant			0..n	XXX	Participant
				RoleCode		1	GDT	EmployeeLeaveRequestParticipantRoleCode
				WorkAgreementID		1	GDT	WorkAgreementID
			Note			0..1	XXX	Note
				Text		1	GDT	Text
BusinessTransactionDocumentReference			LeaveEmployeeTimeReference			0..1	XXX	LeaveEmployeeTimeReference
				ActionCode		1	GDT	ActionCode
				LeaveEmployeeTimeReference		1	GDT	BusinessTransactionDocumentReference
EmployeeTimeItem			LeaveEmployeeTimeItem			0..n	XXX	LeaveEmployeeTimeItem
				CategoryCode		1	GDT	EmployeeTimeItemCategoryCode
				TypeCode		1	GDT	EmployeeTimeItemTypeCode
				Validity		1	GDT	EmployeeTimeItemValidity
				FullWorkingDayIndicator		0..1	GDT	FullWorkingDayIndicator
				Valuation Terms		0..1	XXX	ValuationTerms
					EmployeeTimeValuationTerms	0..1	GDT	EmployeeTimeValuationTerms
				ActivityAllocationAndCostAssignment		0..1	XXX	ActivityAllocationAndCostAssignment
					WorkBreakDownStructureElementID	0..1	GDT	ProjectElementID
				ServiceProvisionAndResourceConsumption		0..1	XXX	ServiceProvisionAndResourceConsumption
					ReceivingCostCentreID	0..1	GDT	CostCentreID
				LogisticConfirmation		0..1	XXX	LogisticConfirmation
					OrderID	0..1	GDT	BusinessTransactionDocumentID
				DifferentPayment		0..1	XXX	DifferentPayment
					PositionID	0..1	GDT	PositionID
				ValuatedDuration		0..n	XXX	ValuatedDuration
					DurationCalculationMethodCode	1	GDT	EmployeeTimeItemDurationCalculationMethodCode
					Duration	1	GDT	Duration
				AdditionalInformation		0..1	XXX	AdditionalInformation
					FirstAdditionalInformationText	0..1	GDT	AdditionalInformationText
					SecondAdditionalInformationText	0..1	GDT	AdditionalInformationText
					ThirdAdditionalInformationText	0..1	GDT	AdditionalInformationText
					FourthAdditionalInformationText	0..1	GDT	AdditionalInformationText
					FifthAdditionalInformationText	0..1	GDT	AdditionalInformationText
					SixthAdditionalInformationText	0..1	GDT	AdditionalInformationText
					SeventhAdditionalInformationText	0..1	GDT	AdditionalInformationText
					EighthAdditionalInformationText	0..1	GDT	AdditionalInformationText
					NinthAdditionalInformationText	0..1	GDT	AdditionalInformationText
					TenthAdditionalInformationText	0..1	GDT	AdditionalInformationText

Level 1 represents the complete message, which (on level two) consists only of the business document object for “EmployeeLeaveRequest”. The message header and log are not required in this case.

The business document consists of four parts. The participant, the note, the employee time reference and the employee time item.

The participant is the employee linked to the business object, which is relevant for the service operation. In the case of the create operation the relevant employee is the requester.

The note is the note from the business object.

The employee time reference is a reference to the existing employee time, which is the subject of this leave request. This reference is only needed if an existing (planned) employee time entry is requested to be changed or canceled.

The employee time item corresponds to the employee leave request item and describes the requested leave.

For all parts of the message the contained data fields are listed together with the global data types with which they are typed.

Proxy Implementation

In order to provide the Web service on the ABAP stack, ABAP entities have to be created with the proxy generator (see Proxy Generation in the ABAP Stack).

Procedure

The message interface in ESR which defines the Web service is,

`EmployeeLeaveRequestCreateRequestConfirmation_In`

Based on this message interface, the proxy generator creates the following ABAP entities:

- ABAP data types
- ABAP Interface: II_TIM_EMPLOYEELR_CRTRC
- ABAP class for the Web service proxy which implements the interface: `CL_TIM_EMPLOYEELR_CRTRC`

The implementation of the proxy is shown in the following in more detail. To keep the example clear, the created ABAP data types are not depicted in this example.

When a Web service call is received, the proxy framework triggers the proxy by calling the `II_TIM_EMPLOYEELR_CRTRC` interface method `EXECUTE_SYNCHRONOUS`. It calls the following methods in the specified order:

1. `INPUT_MAPPING:`Performs the application level mapping of input parameters (see Data Conversion and Custom Modifications)
2. `BUSINESS_LOGIC:`Executes the business logic by calling the existing RFC (see RFC: PT_ESA_REQUEST_CREATE)
3. `OUTPUT_MAPPING:`Performs the application level mapping of output parameters (see Data Conversion and Custom Modifications)

Example

The following is an excerpt of the source code of the EXECUTE_SYNCHRONOUS method.

Execute Synchronous

`METHOD ii_tim_employeelr_crtrc~execute_synchronous.`

Syntax

```
* EmployeeLeaveRequestCreateRequestConfirmation_In DATA
lv_employeenumber TYPE pernr_d.
DATA lv_leave_id       TYPE tim_req_id.
DATA lt_messages       TYPE bapirettab.
DATA ...

* database updates are done synchronously
SET UPDATE TASK LOCAL.

* Input Mapping: map proxy input parameters to function module (RFC) input parameters
CALL METHOD me->input_mapping
EXPORTING
```

```
  is_input = input "/input format according to proxy ABAP datatype
IMPORTING
  ev_employeenumber = lv_employeenumber "/internal format
  ev_leave_id       = lv_leave_id
  ev_operationid    = lv_operationid.

* Business logic call
CALL METHOD me->business_logic
EXPORTING
  iv_employeenumber = lv_employeenumber
  iv_leave_id       = lv_leave_id.
IMPORTING
  ev_request_detail = lv_request_detail       "/result from business logic processing
  ev_success        = lv_success
  et_messages       = lt_messages. "/ application message table -
                                    internal format

* Map application errors table to format of global data type Log
CALL METHOD cl_proxy_log=>fill
EXPORTING
  bapireturn_tab = lt_messages
IMPORTING
  log            = output-... .
EXIT.
ENDIF.

* Output Mapping: map function module (RFC) output parameters to proxy output parameters
CALL METHOD me->output_mapping
EXPORTING
  iv_request_detail = lv_request_detail
IMPORTING
  rs_output = output... . "/ input format according to proxy ABAP
                           datatype

* Commit
cl_soap_commit_rollback=>commit( ).
```

```

ENDMETHOD. "ii_tim_employeelr_crtrc~execute_synchronous

```

End of the code.

The methods called in this excerpt are shown in more detail in the following. The methods for input and output mapping show the mapping of parameters based on a few examples. The method which executes the business logic basically only calls the existing RFC (see RFC: PT_ESA_REQUEST_CREATE)

Input Mapping

```
METHOD input_mapping.
```

Syntax

```
DATA wa_participant TYPE LINE OF hrsef_employeelr_crt_req_msg-employee_leave_request-participant.
DATA lt_messages    TYPE bapirettab.
...

* Mapping employee numbers
LOOP AT is_input-employee_leave_request_create-employee_leave_request-participant INTO wa_participant.
  CASE wa_participant-role_code-value.
    WHEN cl_tim_proxy_constants=>participant_role_owner.
      ev_employeenumber = wa_participant-work_agreement_id-value.
      ...
  ENDCASE.
ENDLOOP.
...

* Mapping LeaveID and ActionCode
CALL METHOD cl_tim_leave_proxy_mappings=>time_ref_input_mapping
EXPORTING
  is_proxy_time_ref  = is_input-employee_leave_request_create-
                       employee_leave_request-
                       leave_employee_time_reference
IMPORTING
  ev_fm_leave_id     = ev_leave_id
  ev_fm_operation_id = ev_operationid
  et_messages        = lt_messages.

```

```
APPEND LINES OF lt_messages TO et_messages.
...

ENDMETHOD. "input_mapping
```

End of the code.

Business Logic

METHOD business_logic.

Syntax

```
DATA : lv_pernr    TYPE pernr_d.
DATA : lt_messages TYPE bapirettab.

lv_pernr  =  iv_employeenumber.

*** If owner pernr is not supplied, use the pernr of the user who is invoking the service
  IF iv_employeenumber IS INITIAL.

    CALL FUNCTION 'PAD_EMPLOYEENUMBERS_GET_LIST'
    EXPORTING
      iv_username        = sy-uname
      iv_key_date        = sy-datum
    IMPORTING
      ev_employee_number = lv_pernr
    TABLES
      t_messages_out     = lt_messages.

    IF NOT lt_messages IS INITIAL.
      APPEND LINES OF lt_messages TO et_messages.
      RETURN.
    ENDIF.

  ENDIF.


CALL FUNCTION 'PT_ESA_REQUEST_CREATE'
EXPORTING
  iv_employee_number      = lv_pernr
  iv_leave_id             = iv_leave_id
  iv_oparation_id         = iv_operationid
  iv_leave_request        = iv_requestdetail
```

```
IMPORTING
  ev_request_detail       = ev_request_detail
  ev_success_confirmation = ev_success
TABLES
  et_messages             = et_messages.

ENDMETHOD.

```

End of the code.

Output Mapping

```
METHOD output_mapping
```

Syntax

```
DATA ls_participant       TYPE ptesa_actor_struc.
DATA ls_proxy_participant TYPE hrsef_employee_leave_request13.
DATA ls_time_item         TYPE hrsef_employee_leave_request10.
DATA lv_owner             TYPE pernr_d.
...

* Participant table
LOOP AT iv_request_detail-participant INTO ls_participant.
  CALL METHOD cl_tim_leave_proxy_mappings=>participant_output_mapping
    EXPORTING
      is_fm_participant    = ls_participant
    IMPORTING
      ev_owner             = lv_owner
      es_proxy_participant = ls_proxy_participant.

  IF NOT ls_proxy_participant IS INITIAL.
    APPEND ls_proxy_participant TO rs_output-
           employee_leave_request_create-employee_leave_request-
           participant.
  ENDIF.
ENDLOOP.
...

* Time item table
CALL METHOD cl_tim_leave_proxy_mappings=>item_output_mapping
```

```
EXPORTING
  iv_pernr      = lv_owner
  is_fm_item    = iv_request_detail-item
IMPORTING
  es_proxy_item = ls_time_item.

APPEND ls_time_item TO rs_output-
employee_leave_request_create-
       employee_leave_request-leave_employee_time_item.

ENDMETHOD. "output_mapping

```

End of the code.

RFC: PT_ESA_REQUEST_CREATE

The RFC used to implement the Web service in the ABAP stack is PT_ESA_REQUEST_CREATE. The RFC not only supports the creation of an employee leave request but also makes it possible to change and delete employee leave requests. Therefore other Web services can be created that are based on the same RFC. As in our example, only employee leave requests are created, the proxy implementing the Web service only uses part of the available functionality.

Basically the RCF takes the employee number, the leave request and an operation type as input. In case of the creation of a request (determined by the operation type) and if executed successfully, it returns the created leave request containing additional detailed information that was added to the original request during creation.

mySAP Service Provisioning: Documentation

The services delivered by SAP for mySAP Business Suite are developed in two steps. First, the services are defined independently of an actual implementation. To do this, they are presented uniformly in a business object model that is valid in general. The models represent the business content of mySAP Business Suite in a simple and structured form.

Note

You can view the models on the Enterprise Service Workplace in SAP Developer Network (SDN) at sdn.sap.com *Enterprise SOA* *ES Workplace*

End of the note.

Second, SAP implements the objects required for realizing the services in the SAP system as ABAP entities.

The uniform and structured view that the business models provide for the functions of mySAP Business Suite is also used for structuring the documentation. Therefore, objects are described in the documentation that do not have a direct correspondent in an SAP system. These consist of the following objects: process components, business objects, and service interfaces. The description of these objects provides you with an overview of the actual business process in which you can use the services delivered by SAP.

The implementations of service operations are available as part of the delivery of mySAP Business Suite enterprise services package.

Documentation Target Groups

The documentation is aimed primarily at the following target groups:

- Business consultants who design processes and applications for customers and who want to learn about the features of the individual services
- Business consultants for service providers who set up a back-end system such that the services can be used in the conventional way
- Software developers who use the services when creating applications (user interfaces) to enable data communication with the back-end system
- Members of the support team who provide support to the above target groups on how to use the service

Documentation Structure

Title

The titles of the documents for the individual SOA entities correspond to the names that were used in the business models to describe the objects.

Note

In following releases, the message interfaces are called operations. For this reason, the message interfaces are now called (service) operations in the documentation.

End of the note.

Technical Data

The technical data of the operations includes the names of the operations as objects in the Enterprise Service Repository (ESR) and specifies what the associated Web service definitions are called.

Documentation

In addition to the technical data, the documentation includes the definitions of the relevant SOA entities as they can be found in the business models. The following sections describe the business use of the objects and their position within the business model. If the data structures of the operation’s message types have special features, these are described in the Requirements, Structure, and Error Handling sections. Information that is valid for all objects of a process component, business object, or service interface is grouped together for each object.

All special features that are based on the implementation of the service and the processing of messages in the SAP back-end system are described in the Notes for SAP Back-End System Administration section.

Features

The following table explains the meaning of the individual entries in the tables of technical data.

Entity Type	Entry in the Table	Meaning
Process component	Software Component	Name of the software component in the SAP back-end system with the release number as of which the implementation of the services of this process component is available
Business object	Software Component	Name of the software component in the SAP back-end system with the release number as of which the implementation of the services of this business object is available
Service interface	Software Component	Name of the software component in the SAP back-end system with the release number as of which the implementation of the services of this service interface is available
	Category	Information indicating whether this is an inbound or outbound interface
	Mode	Technical Name in Enterprise Service Repository (ESR)
Service operation	Technical Name in Enterprise Service Repository (ESR)	Unique name of the operation in ESR
	Namespace in ESR	The namespace indicates the SAP back-end software component in which the specified business logic of the operation is stored.
	Software Component Version in ESR	The software component version indicates the release status of the SAP back-end as of which the operation is available
	Category	Information indicating whether this is an inbound or outbound operation
	Mode	Information indicating whether the operation is used for synchronous or asynchronous processing
	Related Web Service Definition	Name of the Web service definition

Stability of Released Enterprise Services

On the one hand, it is necessary to further develop enterprise services to satisfy new or changed requirements. On the other hand, it is essential that as soon as enterprise services are released, they remain stable. This means they should only be further developed to be upward compatible so that existing consumers do not have to make any adjustments after an upgrade of the enterprise service.

Compatible Further Developments of Enterprise Services

Compatible further developments must necessarily be *syntactically* compatible. This means that the interfaces of a service, especially the XML schema of the input and output message types, may only be enhanced in a particular way. For example, no elements may be renamed, nor may type or length be changed.

A simplified example is only the *syntactically downward-compatible* addition of new optional elements and attributes. An element or attribute is optional if it does not have to be contained in an XML document that is exchanged at runtime. A new optional element or attribute can be added at any place in the interface as a sub-element or attribute of an existing element, as long as that element already has at least one sub-element or attribute.

Note

Since this rule is also valid for response messages, it implies requirements regarding the way enterprise service consumers must process response messages. A consumer must ignore unknown elements or attributes when receiving a response message, meaning they should not strictly validate the messages. In this sense, enterprise services must be *consumer-upward compatible*.

End of the note.

At first glance, additional forms of *syntactically* compatible enhancements would be feasible — especially for input message types. However, they frequently indicate changes that are *incompatible in content* or they are incompatible from the perspective of customer-defined extensions of the data conversion (see BAdIs).

Changes in content arise, for example, if the existing implementation of the service is changed or enhanced, thus causing the service or some of the elements in its interface to receive a modified meaning, without the structure of the interface itself being changed. In addition to the syntactical downward compatibility, *content* compatibility is also essential for service consumers.

If the further development of a service provides for syntactical compatibility as well as content compatibility, the released service itself is enhanced, and a new service is not developed. After an upgrade of the enterprise service, consumer automatically use its enhanced version.

If an incompatible change cannot be avoided, one or more new enterprise services are offered to replace the existing one. However, exceptions to this rule are possible.

New Enterprise Services to Replace Enterprise Services to be Discontinued

- If the incompatible change is a change in content, the new enterprise services will correspondingly have a different name. Thus, the new services must not necessarily have the same name as the replaced services.
- If the change is not a change in *content*, the new enterprise service will essentially keep the name of the replaced service. However, both enterprise services belong to the same namespace, so that the new service cannot have the exact same name. It can be considered to be a new version of the replaced service and therefore has an attached version number "_V1". If there are additional incompatible changes, the version number increases accordingly.

The enterprise service to be replaced is not removed immediately, but rather flagged as “Deprecated” (obsolete). It is still supported in the change release (the release in which the subsequent service is rolled out) and its functionality is retained.

In the subsequent release, the actual implementation of the discontinued service is removed and replaced by coding which merely returns an exception message. At the same time, the enterprise service changes its status to “Revoked.”

The replaced service can only be removed after at least one more release when the discontinuation phase has been entirely finished.

As long as it exists, the discontinued enterprise service refers to its successor services and there is documentation on how existing applications that use the discontinued service can be converted to the new services.

Glossary

Terms used in Enterprise Service Oriented Architecture

A2A

A2A refers to communication between different software applications within the same company. Interfaces for A2A communication, unlike B2B interfaces, may be specific to a certain software application. (Compare B2B)

ABAP

ABAP (Advanced Business Application Programming) is a 4th generation programming language created by SAP. It is currently positioned as the language for programming the SAP Web Application Server, part of the SAP NetWeaver platform for building business applications.

BAPI

A standardized and stable programming interface that facilitates external access to business processes and data in the SAP System. BAPIs offer an object-oriented view of business components in the SAP system.

B2B

B2B refers to communication between different companies. Interfaces for B2B communication must be independent of specific software applications. Usually B2B interfaces comply with international e-business standards. (Compare A2A)

HTTP

Hypertext Transfer Protocol (HTTP) is the protocol used to transfer or convey information on the World Wide Web. It is a patented open internet protocol whose original purpose was to provide a way to publish and receive HTML pages.

More Information

HTTP/1.1 specification (RFC 2616, June 1999)

HTTPS

HTTPS refers to the combination of normal HTTP interaction over an encrypted secure socket layer transport mechanism.

Java

Java refers to a number of computer software products and specifications from Sun Microsystems (the Java™ technology) that together provide a system for developing and deploying cross-platform applications in an object-oriented way.

More Information

http://java.sun.com

RFC

A Remote Function Call is a call of an SAP function module that runs in a different system (destination) from the calling program. Connections are possible between different SAP systems and between an SAP system and a non-SAP system. In non-SAP systems, instead of function modules, special programmed functions are called, whose interface simulates a function module. The term RFC sometimes also refers to the called function module.

SAML

Security Assertion Markup Language (SAML), developed by the Security Services Technical Committee of OASIS, is an XML-based framework for communicating user authentication, entitlement, and attribute information. SAML allows business entities to make assertions regarding the identity, attributes, and entitlements of a subject (an entity that is often a human user) to other entities, such as a partner company or another enterprise application.

More Information

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

SSL

Secure Sockets Layer (SSL) is a cryptographic protocol which provides secure communication on the Internet.

XML

Extended Markup Language (XML) is a generic language for defining new markup languages specified by the World Wide Web Consortium (W3C). Markup languages (such as HTML) are used to represent documents with a nested, treelike structure.

More Information

http://www.w3.org/XML/

XML Encryption

XML Encryption is a specification that defines how to encrypt the content of an XML element, as recommended by the World Wide Web Consortium (W3C).

More Information

http://www.w3.org/Encryption/2001/

XML Signature

XML Signature is a specification that defines an XML syntax for digital signatures, as recommended by the World Wide Web Consortium (W3C).

More Information

http://www.w3.org/TR/xmldsig-core/

About This Documentation

Documentation Target Groups

The documentation is aimed primarily at the following target groups:

- Business consultants who design processes and applications for customers and who want to learn about the features of the individual services
- Business consultants for service providers who set up a back-end system such that the services can be used in the conventional way

- Software developers who use the services when creating applications (user interfaces) to enable data communication with the back-end system
- Members of the support team who provide support to the above target groups on how to use the service

Documentation Structure

Title

The titles of the documents for the individual SOA entities correspond to the names that were used in the business models to describe the objects.

In following releases, the *message interfaces* are called *operations*. For this reason, the *message interfaces* are now called *(service) operations* in the documentation.

Technical Data

The technical data of the operations includes the names of the operations as objects in the *Enterprise Service Repository* (ESR) and specifies what the associated Web service definitions are called.

Documentation

In addition to the technical data, the documentation includes the definitions of the relevant SOA entities as they can be found in the business models. The following sections describe the business use of the objects and their position within the business model. If the data structures of the operation's message types have special features, these are described in the Requirements, Structure, and Error Handling sections. Information that is valid for all objects of a process component, business object, or service interface is grouped together for each object.

All special features that are based on the implementation of the service and the processing of messages in the SAP back-end system are described in the *Notes for SAP Back-End System Administration* section.

Tables of Technical Data

The following tables explain the meaning of the individual entries in the tables of technical data.

a. Entity Type	b. Entry in the Table	c. Meaning
d. Process component	e. Software Component	f. Name of the software component in the SAP back-end system with the release number as of which the implementation of the services of this process component is available
g. Business object	h. Software Component	i. Name of the software component in the SAP back-end system with the release number

		as of which the implementation of the services of this business object is available
j. Service interface	k. Software Component	l. Name of the software component in the SAP back-end system with the release number as of which the implementation of the services of this service interface is available
m.	n. Category	o. Information indicating whether this is an inbound or outbound interface
p.	q. Mode	r. Information indicating whether the interface supports synchronous or asynchronous processing
s. Service operation	t. Technical Name in Enterprise Service Repository (ESR)	u. Unique name of the operation in ESR
v.	w. Namespace in ESR	x. The namespace indicates the SAP back-end software component in which the specified business logic of the operation is stored.
y.	z. Software Component Version in ESR	aa. The software component version indicates the release status of the SAP back-end as of which the operation is available.

bb.	cc. Category	dd. Information indicating whether this is an inbound or outbound operation
ee.	ff. Mode	gg. Information indicating whether the operation is used for synchronous or asynchronous processing
hh.	ii. Related Web Service Definition	jj. Name of the Web service definition

SAP: Important Disclaimers and Legal Information

The services delivered by SAP for *mySAP Business Suite* are developed in two steps. First, the services are defined independent of an actual implementation. To do this, they are presented uniformly in a business object model that is valid in general. The models represent the business content of *mySAP Business Suite* in a simple and structured form.

You can view the models on the *Enterprise Service Workplace* in *SAP Developer Network* (SDN) at **`sdn.sap.com`** → **`Enterprise SOA`** → **`ES Workplace`**.

Second, SAP implements the objects required for realizing the services in the SAP system as ABAP entities.

The uniform and structured view that the business models provide for the functions of *mySAP Business Suite* is also used for structuring the documentation. Therefore, objects are described in the documentation that do not have a direct correspondent in an SAP system. These consist of the following objects: process components, business objects, and service interfaces. The description of these objects provides you with an overview of the actual business process in which you can use the services delivered by SAP.

The implementations of service operations are available as part of the delivery of *mySAP Business Suite enterprise services package*.

Business Partner Data Processing

Technical Data

Entity Type	Process component
Software Component	BBPCRM

Definition

Enables a company to manage all business partner data that is relevant to control its business processes, such as sales, purchasing and accounting processes.

Use

This process component facilitates the creation and maintenance of business partner data by means of core services. It enables you to provide lean applications designed for various situations in which basic functions are required for processing business partner data.

The service operations available support the following typical scenarios:

- Searching for business partners based on various criteria
- Creating business partners with required address data and telephone number
- Identifying a customer and updating basic data, for example, recording a change of address
- Creating new contact person data, for example, after business events and meetings
- Identifying customers within a marketing campaign based on various criteria, for example, a response ID, and updating information on their preferences and interests

This data is integrated in the relevant business processes in the back-end system.

Notes for SAP Back-End System Administration

Prerequisites

To use the operations contained in this process component, you have to be using the application component *Business Partners* (CRM-MD-BP), and for the marketing attributes *Marketing* (CRM-MKT).

Processing

Take into account the following behavior of the change and update operations of this process component when you use it:

The data transferred by request messages overwrites only those affected objects in the databases of the back-end system that are also filled in the message structure.

This means that if optional elements are not filled in a request message, then the back-end system leaves the values as they are in the affected fields in the database. The only exception to this behavior is when these values are automatically updated due to changes to other data.

However, for list-type elements that can contain several items, data transferred completely overwrites the corresponding objects in the database of the back-end system. The response of the back-end system, therefore, is to expect a value for all items of these elements. You can recognize these list-type elements in that *unbounded* is entered for maxOccurs.

This means that if items of list-type elements are not filled in a request message, the back-end system interprets this information as a deletion entry for the item concerned, and initializes the corresponding fields in the database.

An order contains items 10, 20, 30, 40, 50. With the request message for a change operation, item 10 is to be changed, and item 30 is to be deleted.

So that you do not delete items 20, 40, and 50 with the change operation, you also have to specify items 20, 40, and 50 even if they have not been changed, in addition to items 10 and 30. Item 30 is deleted if you do not specify it.

Business Partner

Technical Data

Entity Type	Business object
Software Component	BBPCRM

Definition

A business partner is a person, an organization, or a group of persons or organizations in which a company has a business interest.

Use

A business partner comprises general data that is not directly related to a business process. This includes data such as roles, relationships, addresses, ID numbers, and bank details.

A business partner also contains data that is required for specific business processes. This includes sales data and purchasing data.

Business partner data is integrated in a large variety of transactional contexts, such as sales orders, service orders, and marketing campaigns.

Constraints

This business object provides service operations enabling business partners to be created and maintained with their basic data only. Within this scope, a business partner comprises general data (such as name and address) and the business partner relationship “contact person”.

Query Business Partner

Technical Data

Entity Type	Service interface
Software Component	BBPCRM
Category	Inbound
Mode	Synchronous

Definition

Group of operations used to retrieve a list of business partner records based on different criteria and provide a list of business partner records as a result.

Use

This inbound service interface can be used to search for business partners based on various criteria, such as basic data (name and address) and relationship type.

Integration

The inbound service interface *Query Business Partner* belongs to the process component Business Partner Data Processing [Seite 78] and business object Business Partner [Seite 80].

Structure

The service interface groups the following inbound operations:

- BusinessPartnerBasicDataByCommunicationQueryResponse (Find Business Partner by Communication [Seite 81])
- BusinessPartnerBasicDataByNameAndAddressQueryResponse (Find Business Partner by Name and Address [Seite 82])
- BusinessPartnerBasicDataByResponseIDQueryResponse (Find Business Partner by Response ID [Seite 83])
- BusinessPartnerContactPersonByNameAndWorkplaceQueryResponse (Find Contact Person by Name and Workplace [Seite 84])

Error Handling

Each service operation has a log structure that lists any errors that occur.

Find Business Partner by Communication

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	BusinessPartnerBasicDataByCommunicationQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_BPBASICDATABYCOMQR

Definition

Retrieves a list of business partner records based on address-dependent or address-independent communication data.

Use

This inbound service operation can be used to search for an existing business partner based on communication data (address-dependent and address-independent).

It can be used within a duplicate check, enabling an existing company or individual to be identified based on communication data.

For example, when creating new contact data, the duplicate check can be performed against the e-mail address of the contact person, and, if no duplicates are found, the new contact person created in the system.

Features

- The operation searches for all business partner addresses as well as address-independent communication data.
- The operation returns the name of the business partner and the standard address.
- It also returns the standard phone, mobile phone, fax, and e-mail.
- Address identifiers are returned in cases where the identification resulted in a non-standard address. Additional address details can be found by calling the service operation BusinessPartnerBasicDataByIDQueryResponse.
- The operation processes the following message types:
    - BusinessPartnerBasicDataByCommunicationQuery_sync
    - BusinessPartnerBasicDataByCommunicationResponse_sync

Find Business Partner by Name and Address

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	BusinessPartnerBasicDataByNameAndAddressQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_BPBASICDATABYNAMEADDRQR

Definition

Retrieves a list of business partner records based on basic data (a given name and address attributes).

Use

This inbound service operation is used to search for business partners based on name and address data. It also provides additional filtering on the category and role, and displays only the standard address.

This service operation is primarily used to identify whether a specific business partner already exists in the system or whether it needs to be created.

Features

- The query is executed based on the name, address information, and role (filtering of the search result based on one or more given roles.)
- The information returned includes name, roles, standard address, standard phone, mobile number, fax, and e-mail.
- A list of address identifiers is returned in cases when the identification resulted in one or several non-standard addresses. Additional address details can be found by calling the service operation BusinessPartnerBasicDataByIDQueryResponse.
- This inbound service operation processes the following message types:
  - BusinessPartnerBasicDataByNameAndAddressQuery_sync
  - BusinessPartnerBasicDataByNameAndAddressResponse_sync

Find Business Partner by Response ID

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	BusinessPartnerBasicDataByResponseIDQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_BPBASICDATABYRESPONSIDQR

Definition

Retrieves a list of business partner records based on a given response code.

Use

Used within a marketing context, this inbound service operation allows a business partner to be identified based on a response code. As part of a marketing campaign, for example, response codes may be received from sales prospects by different channels and used as the basis for identifying sales prospects and contact persons in the system, thus enabling their data to be retrieved and follow-up processes triggered.

Features

- The information returned includes name, standard address, standard phone, mobile phone, fax, and e-mail.
- The operation processes the following message types:
  - BusinessPartnerBasicDataByResponseIDQuery_sync

- BusinessPartnerBasicDataByResponseIDResponse_sync

Find Contact Person by Name and Workplace

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	BusinessPartnerContactPersonByNameAndWorkplaceQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_BPCONBYNAMEWORKPLACEQR

Definition

Retrieves a list of contact person relationships based on the search criteria specified concerning a contact person, such as name, workplace address, and communication data of the workplace address.

Use

This inbound service operation is used to search for contact person relationships based on the search criteria specified for the contact person.

It is primarily used to identify whether a specific contact person already exists in the system or whether it needs to be created.

Features

- The query is executed based on the contact person relationship, contact person name, communication data, business partner ID and business partner name.
- If communication data is specified, the search is based on this data, this being the most specific information.

  The communication data may be part of the contact person’s workplace address, the company address, or the private address of the individual concerned. The search analyzes all three locations.
- The key information returned comprises the IDs of the company and contact person.
- Other information returned includes the name of the person, standard workplace address, standard phone, mobile phone, fax, and e-mail.

- Other address identifiers are returned in cases when the identification resulted in a non-standard workplace address. Additional address details can be found by calling the service operation BusinessPartnerBasicDataByIDQueryResponse.
- The only information returned about the business partner is the business patner ID. Further details have to be read using BusinessPartnerBasicDataByIDQueryResponse.
- The service operation processes the following message types:
  - BusinessPartnerContactPersonByNameAndWorkplaceQuery_sync
  - BusinessPartnerContactPersonByNameAndWorkplaceResponse_sync

Manage Business Partner

Technical Data

Entity Type	Service interface
Software Component	BBPCRM
Category	Inbound
Mode	Synchronous

Definition

Group of operations used to manage business partner data.

Use

This inbound service interface can be used to create and change business partner basic data, such as name and address, and relationships, enabling customer and contact data to be created and kept up-to-date.

Integration

The inbound service interface *Manage Business Partner* belongs to the process component Business Partner Data Processing [Seite 78] and business object Business Partner [Seite 80].

Structure

The service interface groups the following inbound operations:

- BusinessPartnerBasicDataCreateRequestConfirmation (Create Business Partner with Basic Data [Seite 86])
- BusinessPartnerBasicDataChangeRequestConfirmation (Change Business Partner Basic Data [Seite 87])
- BusinessPartnerContactPersonCreateRequestConfirmation (Create Business Partner Contact Person [Seite 88])
- BusinessPartnerContactPersonChangeRequestConfirmation (Change Business Partner Contact Person [Seite 89])
- BusinessPartnerBasicDataByIDQueryResponse (Read Business Partner Basic Data [Seite 90])

Error Handling

Each service operation has a log structure that lists any errors that occur.

Create Business Partner with Basic Data

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	BusinessPartnerBasicDataCreateRequestConfirmation_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_BPBASICDATACRTRC

Definition

Creates a business partner record with its basic data, this being name and address.

Use

This inbound service operation is a basic operation that is designed for creating business partner data in a variety of senarios, such as Internet self-service and field sales.

Features

- The service operation creates the name of the business partner and standard address. It also creates the standard phone, mobile phone, fax, and e-mail.
- When creating new basic data, the address part is restricted to one address, the standard address, plus the standard communication data of type phone, fax, e-mail, mobile phone, and URL (home page). It is not possible to create multiple address versions.
- In addition to the name and address parameters, one or more roles can be passed which are assigned to the new business partner from the very start.
- The operation processes the following message types:
  - BusinessPartnerBasicDataCreateRequest_sync
  - BusinessPartnerBasicDataCreateConfirmation_sync

Error Handling

A list of error messages as part of the message log and optionally a list of possible address duplicates.

A control parameter allows a duplicate check to be triggered or suppressed. If the duplicate check is executed and duplicates found, the create operation may be interrupted, depending on the control parameter value.

The AddressDuplicateCheckExecutionMethodCode is a fixed SAP code list:

Code	Name	Description
1	No check	No check
2	Check, no interrupt	Check, no interrupt in case of duplicates
3	Check and interrupt	Check and interrupt in case of duplicates

Change Business Partner Basic Data

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	BusinessPartnerBasicDataChangeRequestConfirmation_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_BPBASICDATACHGRC

Definition

Changes the basic data (name and address) of a business partner.

Use

This inbound service operation can be made available in scenarios where it is important that customers' basic data is up-to-date. Particularly after a change of address, this enables customers to update their name and address data themselves.

For example, due to an address change a letter does not reach a customer and is returned. A sales assistant provided with a lean application designed specifically for handling this type of situation is able to search for and identify the customer involved and contact them by e-mail. By using the service operation within an interactive form, for example, it can be included in the e-mail enabling the customer to easily enter their new data, which is subsequently updated in the back-end system.

Features

- The service operation changes the name and address (currently valid) of the business partner, and the standard communication data of phone, fax, e-mail, mobile phone, and URL.
- The service operation is not restricted to the standard address, but applies to any address specified by an address key. The default is the standard address.
- The operation processes the following message types:

- BusinessPartnerBasicDataChangeRequest_sync
- BusinessPartnerBasicDataChangeConfirmation_sync

Error Handling

A list of error messages as part of the message log and optionally a list of possible address duplicates.

A control parameter allows a duplicate check to be triggered or suppressed. If the duplicate check is executed and duplicates found, the change operation may be interrupted, depending on the control parameter value.

The AddressDuplicateCheckExecutionMethodCode is a fixed SAP code list:

Code	Name	Description
1	No check	No check
2	Check, no interrupt	Check, no interrupt in case of duplicates
3	Check and interrupt	Check and interrupt in case of duplicates

Constraints

Note that time dependency is not available. Thus a change of address can only be recorded to the extent that the new address is entered in the system, a history of previous addresses is not, however, available.

Create Business Partner Contact Person

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	BusinessPartnerContactPersonCreateRequestConfirmation_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_BPCONTACTPERSONCRTRC

Definition

Creates a business partner relationship of type contact person.

Use

This inbound service operation can be used to create contact person data, enabling, for example, sales representatives to enter contact data quickly and easily, such as after trade fairs and offsite meetings, where new contacts may have just been made and business card information needs to be entered in the system as soon as possible.

If implemented appropriately in a particular application, contact data could first be entered offline, and then later created automatically in the back-end system, provided the data consistency check does not return any errors.

Features

- The service operation creates a contact person relationship and implicitly a business partner record of type person with name, address, and standard communication.
- It creates the relationship attributes, such as department and function.
- It assigns a standard workplace address, allowing the standard communication data of the workplace address (standard phone, mobile, fax number, and e-mail) to be specified.
- The operation processes the following message types:
    - BusinessPartnerContactPersonCreateRequest_sync
    - BusinessPartnerContactPersonCreateConfirmation_sync

Prerequisites

The organization record has to have been created beforehand, for example, by BusinessPartnerBasicDataCreateRequest.

Change Business Partner Contact Person

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	BusinessPartnerContactPersonChangeRequestConfirmation_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_BPCONTACTPERSONCHGRC

Definition

Changes the attributes of a business partner relationship of type contact person.

Use

This inbound service operation is designed for use in situations where contact data needs to be corrected or updated.

The contact person ID is known and the contact person already exists in the back-end system.

Features

- The service operation changes the attributes of the contact person relationship, such as address, telephone number, fax.
- It can assign a different standard workplace address.
- It can change the standard communication data of the workplace address (phone, fax, mobile phone, e-mail), or create them for a newly assigned workplace address.
- It cannot change data such as name or private address, nor assign a different company address. This is covered by BusinessPartnerBasicDataChangeRequest.
- The operation processes the following message types:
  - BusinessPartnerContactPersonChangeRequest_sync
  - BusinessPartnerContactPersonChangeConfirmation_sync

Read Business Partner Basic Data

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	BusinessPartnerBasicDataByIDQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_BPBASICDATABYIDQR

Definition

Reads the basic data of a business partner, this being the name and address.

Use

This inbound service operation can be used to retrieve business partner details, when the ID is known.

It is suitable for use after a duplicate check, for example, to display the details of the specific business partner found to check whether this data should be changed or a new business partner created.

In the marketing context, for example, marketing data may be required for a campaign. After a query, the address can be identified for a particular customer and their data then retrieved for use in further processes.

Features

- The query is executed based on the business partner ID and address ID.
- The information returned includes name, standard address, standard phone, mobile number, fax, and e-mail.
- The operation processes the following message types:
  - BusinessPartnerBasicDataByIDQuery_sync
  - BusinessPartnerBasicDataByIDResponse_sync

Read Business Partner Contact Person

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	BusinessPartnerContactPersonByRelationshipIDQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_BPCONBYRELATIONSHIPIDQR

Definition

Reads the business partner relationship of type contact person for a given business partner and contact person.

Use

This inbound service operation is used to retrieve the data maintained for a specified contact person, enabling it to be used by other business transactions, such as sales orders, and integrated in further processes.

Features

- The query is executed based on the business partner ID and contact person ID.
- The service operation returns the name of the person, attributes of the contact person relationship, the standard workplace address, and the standard communication data of the workplace address.
- The only information returned about the business partner is the business partner ID. Further details have to be read using BusinessPartnerBasicDataByIDQueryResponse.

- The operation processes the following message types:
  - BusinessPartnerContactPersonByRelationshipIDQuery_sync
  - BusinessPartnerContactPersonByRelationshipIDResponse_sync

Customer

Technical Data

Entity Type	Business object
Software Component	BBPCRM

Definition

A customer is a business partner with whom a business relationship exists. Customers can be internal or external, and they receive, use, and pay for goods or services that are provided by a vendor or service provider.

Use

A customer is used within processes involving the sales of products and services. A customer can assume all functions that a business partner performs in sales-related business processes.

Besides the general business partner data, such as addresses, roles, relationships, and bank details, the business object provides all data required for the following processes:

- Customer contact

  This includes the following information:
  - The times when a business partner can be contacted (a customer's visiting hours, calling hours, and goods receiving hours)
  - The times when a business partner's contact person can be contacted (a contact person's visiting hours and calling hours)
- Capital investment

  This includes attributes of the shareholder relationship, such as capital amount and the percentage of the investment
- Marketing processes

  This includes attributes that are relevant for marketing processes (for example, Nielsen ID)

Irrespective of the business processes in which the customer is involved, they can assume roles such as prospect and bill-to party.

The business object *Customer* is derived from the business object Business Partner [Seite 80].

Manage Customer

Technical Data

Entity Type	Service interface
Software Component	BBPCRM
Category	Inbound
Mode	Synchronous

Definition

Group of operations used to manage customer data.

Use

This inbound service interface can be used to maintain the assignment of marketing attributes to a customer, thus enabling marketing information to be recorded for the customer and updated when required.

Prerequisites

The customer (business partner) already exists in the system.

The marketing attributes have been maintained in the system.

Integration

The inbound service interface *Manage Customer* belongs to the process component Business Partner Data Processing [Seite 78] and business object Customer [Seite 92].

Structure

The service interface groups the following inbound operations:

- CustomerMarketingAttributesCreateRequestConfirmation (Create Customer Marketing Attributes [Seite 93])
- CustomerMarketingAttributesChangeRequestConfirmation (Change Customer Marketing Attributes [Seite 94])
- CustomerMarketingAttributesByBusinessPartnerQueryResponse (Read Customer Marketing Attributes [Seite 95])

Error Handling

Each service operation has a log structure that lists any errors that occur.

Create Customer Marketing Attributes

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	CustomerMarketingAttributesCreateRequestConfirmation_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1

Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_CUSTOMERMKTATTRCRTRC

Definition

Creates marketing attributes for a business partner.

Use

This inbound service operation can be used within the marketing context to create marketing attributes for a customer or potential customer. The marketing attributes as such already exist in the system. The operation assigns the specified values to the attributes within the attribute set for the customer involved, thus enabling the information required, as defined in Customizing, to be recorded for the customer.

Features

- The customer number (business partner ID), marketing attribute set, marketing attributes, and relevant values are entered (in tabular form).
- The service operation creates the value assignment for the set of marketing attributes in the system.
- If created successfully, this is recorded in the log only.
- This operation processes the following message types:
    - CustomerMarketingAttributesCreateRequest_sync
    - CustomerMarketingAttributesCreateConfirmation_sync

Prerequisites

The marketing attributes must already be defined in the system, but the marketing set not yet assigned to the customer, otherwise processing is rejected.

Change Customer Marketing Attributes

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	CustomerMarketingAttributesChangeRequestConfirmation_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous

Related Web Service Definition	CRM_CUSTOMERMKTATTRCHGRC

Definition

Changes marketing attributes for a business partner.

Use

This inbound service operation can be used within the marketing context to change marketing attributes for a customer or potential customer. The operation changes the values assigned to the attribute set already created for the customer.

Features

- The customer number (business partner ID), marketing attribute set, marketing attributes, and relevant values are entered (in tabular form).
- The operation does not only change single-value attributes, but can add additional values within the attribute set.
- The operation processes the following message types:
  - CustomerMarketingAttributesChangeRequest_sync
  - CustomerMarketingAttributesChangeConfirmation_sync

Prerequisites

A marketing attribute set must already have been assigned to the customer.

Read Customer Marketing Attributes

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	CustomerMarketingAttributesByBusinessPartnerQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_CUSTOMERMKTATTRBYBPQR

Definition

Reads the marketing attributes of a business partner.

Use

This inbound service operation can be used to retrieve marketing information about a particular customer or potential customer. It enables the full set of marketing attributes assigned to the customer to be made available.

Features

- The query is executed for the customer number (business partner ID).
- The information returned includes the business partner ID and the assigned attributes and their values, in tabular form.
- In addition, it also returns in tabular form the attributes that have not been assigned but are allowed, and their possible values.
- The operation processes the following message types:
    - CustomerMarketingAttributesByBusinessPartnerQuery_sync
    - CustomerMarketingAttributesByBusinessPartnerResponse_sync

Prerequisites

The customer must exist in the system and a marketing attribute set must already have been assigned.

Customer Quote Processing

Technical Data

Entity Type	Process component
Software Component	BBPCRM

Definition

Provides and manages an offer made to a customer for the delivery of goods or services according to fixed terms. The offer legally binds the company for a certain period of time.

Use

This process component enables you to provide lean applications designed for use within the quotation approval process. It provides the basic functions required within a typical scenario:

- After negotiation with a customer or prospect, a sales representative can create a customer quote with its basic data.
- If the information provided the first time was not complete or not entirely correct, the sales representative can access the quote again to make the necessary changes.
- Once the customer quote data is correct and is sent to the customer or prospect, they can either accept or reject the quote, enabling follow-up processes involved in quotation processing to be triggered in the back-end system.

Notes for the SAP Back-End System Administration

Prerequisites

To use the operations contained in this process component, you have to be using the application components *Customer Inquiries / Quotations* (CRM-BTX-SLO-QUT) and *Business Partners* (CRM-MD-BP).

Processing

Take into account the following behavior of the change and update operations of this process component when you use it:

The data transferred by the request messages completely overwrites the corresponding objects in the database of the back-end system. The response of the back-end system, therefore, is to expect a value for all elements that are found in the message type structure. This also applies to elements that are designated as optional in the message structure.

> This means that if elements are not filled in a request message, the back-end system interprets this information as a deletion entry, and initializes the corresponding fields in the database.
>
> For list-type elements that can contain several items, the items that are not filled are interpreted as a deletion entry. You can recognize these list-type elements in that *unbounded* is entered for maxOccurs.

> An order contains items 10, 20, 30, 40, 50. With the request message for a change operation, item 10 is to be changed, and item 30 is to be deleted.
>
> So that you do not delete items 20, 40, and 50 with the change operation, you also have to specify items 20, 40, and 50 in addition to items 10 and 30, even if the former have not been changed. Item 30 is deleted if you do not specify it.

Performing a read directly before sending the change request ensures that the change request contains the complete data record. When using change operations, it is important that a read is always performed beforehand, otherwise you may lose data.

Customer Quote

Technical Data

Entity Type	Business object
Software Component	BBPCRM

Definition

A customer quote is an offer from a seller to a customer regarding the delivery of goods and provision of services at a certain prices, for a certain quantity, and at a certain time. The offer is binding for the seller for a certain time period.

Use

A customer quote comprises general information that applies to the whole customer quote, such as the parties involved, sales and delivery agreements, status and references.

Other information is available at item and schedule line level. The items of the customer quote provide item information and dependent data such as product information, the parties involved, the agreements on sales and delivery, and the status and references. The schedule lines of an item specify when products are to be made available and in what quantity.

Constraints

This business object provides service operations enabling a lean form of quotation management. This means that the data processing options are restricted to the essential core operations required, and the data available for processing has been considerably restricted.

Notes for SAP Back-End System Administration

Constraints

The customer quote provides the following data:

- Quote ID

  It does not provide the description.
- Party ID (customer number)

  This is restricted to the buyer party and is the global data type *PartyPartyID*.
- Validity period (begin and end)
- Total value

  This is the net amount (with the corresponding currency unit). It does not include tax or gross value.

The item provides the following information:

- Item ID
- Description
- Product (seller ID)

  This is the global data type *ProductPartyID*.

  Products are:
  - Simple
  - Materials (not services)
  - Not configurable
- Quantity
- Total value

Manage Customer Quote

Technical Data

Entity Type	Service interface
Software Component	BBPCRM
Category	Inbound
Mode	Synchronous

Definition

Group of operations for managing customer quote data.

Use

This inbound service interface provides core service operations for creating and changing customer quote data.

Integration

The inbound service interface *Manage Customer Quote* belongs to the process component Customer Quote Processing [Seite 96] and business object Customer Quote [Seite 97].

Structure

The service interface groups the following inbound operations:

- CustomerQuoteCreateRequestConfirmation (Create Customer Quote [Seite 99])
- CustomerQuoteChangeRequestConfirmation (Change Customer Quote [Seite 100])
- CustomerQuoteByIDQueryResponse (Read Customer Quote [Seite 101])

Error Handling

Each service operation has a log structure that lists any errors that occur.

Notes for SAP Back-End System Administration

Prerequisites

Although standard Customizing for quotations is required to work with these operations, the Customizing for partner determination has to be modified.

In standard Customizing, an employee responsible is mandatory for partner determination in quotation processing. These service operations, however, do not require an employee and do not work if an employee responsible has been assigned in the partner determination procedure.

It is therefore important that you set up a new partner determination procedure and explicitly remove the employee.

Create Customer Quote

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	CustomerQuoteCreateRequestConfirmation_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_CUSTQTCRTRC

Definition

Creates a customer quote using the information about the customer and the products needed.

Use

This inbound service operation enables a customer quote to be created quickly and easily. It provides only a selection of the data normally available in full quote creation and thus enables a quote to be created with minimal data.

Features

- The customer number (party ID) and item ID are specified, as well as the product and quantity. Other information is generated based on Customizing in the back-end system.
- The operation provides all basic quotation data required, this being buyer party, product, quantity, valid from and to date (both must be specified), and total value (net amount).
- The ID is returned as confirmation.
- The operation processes the following message types:
  - CustomerQuoteCreateRequest_sync
  - CustomerQuoteCreateConfirmation_sync

Prerequisites

The customer number and product ID must be known.

Change Customer Quote

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	CustomerQuoteChangeRequestConfirmation_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_CUSTQTCHGRC

Definition

Changes the data of a customer quote.

Use

This inbound service operation can be used to change customer quotes already created in the system. It provides only a selection of the data normally available and is designed primarily to enable the user to respond to any errors that might occur and thus correct the necessary item information, this being product ID or quantity. The buyer party can also be changed if the customer number does not yet exist in the system.

Features

- The customer number and quote ID are entered.
- The operation changes only the product ID and quantity.
- The operation provides all required quotation data, such as transaction ID, business partner, product, quantity, valid from and to date, pricing (net value).
- The operation processes the following message types:
    - CustomerQuoteChangeRequest_sync
    - CustomerQuoteChangeConfirmation_sync

Prerequisites

The customer number and product ID must be known.

Read Customer Quote

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	CustomerQuoteByIDQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_CUSTQTBYIDQR

Definition

An inquiry for a customer quote by its ID. The response contains the details of the offer made to the customer.

Use

This inbound service operation enables the details of a customer quote to be retrieved. Unlike the create, which returns only the quote ID as confirmation, this operation provides the full details of the offer made to the customer.

Features

- The customer number and quote ID are entered.
- The service operation provides all required quotation data, this being quote ID, party ID, product, quantity, valid from and to date, pricing (net value).
- It does not return the buyer name.

- The operation processes the following message types:
  - CustomerQuoteByIDQuery_sync
  - CustomerQuoteByIDResponse_sync

Prerequisites

The customer number and quote ID must be known.

Customer Quote Action

Technical Data

Entity Type	Service interface
Software Component	BBPCRM
Category	Inbound
Mode	Synchronous

Definition

Service interface containing an operation for recording a customer's action regarding a specific quote and setting the status accordingly in the system.

Use

By enabling a customer's action to be recorded regarding a specific quote and setting the status in the system accordingly, this inbound service interface enables follow-up processes involved in quotation processing to be triggered. As such, it is not used to create or change data, but rather to confirm a customer's action by setting the appropriate status in the system, and thus to control further processing.

Integration

The inbound service interface *Manage Customer Quote* belongs to the process component Customer Quote Processing [Seite 96] and business object Customer Quote [Seite 97].

Structure

The service interface contains the inbound operation CustomerQuoteAcceptanceAcknowledgementRequestConfirmation (Acceptance Acknowledgement Customer Quote [Seite 102]).

Error Handling

Each service operation has a log structure that lists any errors that occur.

Acceptance Acknowledgement Customer Quote

Technical Data

Entity Type	Service operation

Technical Name in Enterprise Service Repository (ESR)	CustomerQuoteAcceptanceAcknowledgementRequestConfirmation_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_CUSTQTACCEPTACKRC

Definition

Request from a customer to *Customer Quote Processing* to acknowledge his quote acceptance and the confirmation containing the customer quote ID.The response indicates if the customer's acceptance of the customer quote has been acknowledged in *Customer Quote Processing*.

Use

This inbound service interface records a customer's acceptance of a specific quote and sets the appropriate status in the system. It confirms this action by returning the customer quote ID.

In a typical scenario, for example, a customer quote is sent by e-mail to the customer for approval. Depending on the customer's action, the system either sets the status of the quote to "Accepted", which triggers follow-up activities such as the creation of a sales order, or forwards it to the sales representative to deal with the open issues and restart the process

Features

- The quote ID and acceptance status code are sent for confirmation.
- The acceptance status code can have the following values:
  - AP: Accepted – quote has been accepted
  - AJ: Pending – a decision has not yet been made
  - RE: Rejected – quote has been rejected
- Depending on Customizing in the back-end system, accepting the quote can entail the following:
  - Accept with no follow-up action
  - Accept by triggering a follow-up document
- The operation processes the following message types:
  - CustomerQuoteAcceptanceAcknowledgementRequest_sync
  - CustomerQuoteAcceptanceAcknowledgementConfirmation_sync

Lead Processing

Technical Data

Entity Type	Process component
Software Component	BBPCRM

Definition

Handling of the potential interest of a business partner and the interactions with him over a certain time frame.

Use

This process component enables you to provide lean applications designed for creating and processing lead data.

The quick creation of leads can be used to support and fulfill multiple processes around lead processing. This may include functions such as:

- Searching for and reading lead data based on various criteria
- Creating leads with their basic data, this being lead description, classification data, qualification data, and priority
- Assigning products and other data, such as notes

Notes for SAP Back-End Administration

Prerequisites

To use the operations contained in this process component, you have to be using the application component *Lead Management* (CRM-BTX-LEA), *Business Partners* (CRM-MD-BP), *Marketing* (CRM-MKT).

Processing

Take into account the following behavior of the change and update operations of this process component when you use it:

The data transferred by the request messages completely overwrites the corresponding objects in the database of the back-end system. The response of the back-end system, therefore, is to expect a value for all elements that are found in the message type structure. This also applies to elements that are designated as optional in the message structure.

This means that if elements are not filled in a request message, the back-end system interprets this information as a deletion entry, and initializes the corresponding fields in the database.

For list-type elements that can contain several items, the items that are not filled are interpreted as a deletion entry. You can recognize these list-type elements in that *unbounded* is entered for maxOccurs.

An order contains items 10, 20, 30, 40, 50. With the request message for a change operation, item 10 is to be changed, and item 30 is to be deleted.

So that you do not delete items 20, 40, and 50 with the change operation, you also have to specify items 20, 40, and 50 in addition to items 10 and 30, even if the former have not been changed. Item 30 is deleted if you do not specify it.

Performing a read directly before sending the change request ensures that the change request contains the complete data record. When using change operations, it is important that a read is always performed beforehand, otherwise you may lose data.

Lead

Technical Data

Entity Type	Business object
Software Component	BBPCRM

Definition

A business transaction that describes the potential or projected business interests of a business partner and the interactions based on this, over a period of time.

It contains information for uniquely identifying it, and business-relevant entries, such as the priority, origin, and category of a lead. It also contains the party that the lead is based on and the parties involved in implementing the lead.

Use

A lead contains information used as the basis for making decisions about generating an opportunity.

A lead comprises general data such as parties involved, dates, durations, status, attachments/texts, references, that are valid for the complete object, and item information necessary for the lead and that apply for it, such as product information, product category, quantity, unit of measure.

A lead can be described in more detail by:

- An opportunity/lead origin type that describes the origin (source of generation) of the lead
- A grouping used for evaluation purposes
- A priority that describes its urgency
- A (manually or automatically assigned) lead qualification level that describes the potential/projected interest of the business partner

Constraints

This business object provides only core service operations for lead processing. The full range of data is not available.

Surveys are not available, nor can attachments be created, modified, or deleted. It is not possible to change the status nor references to activities or opportunities.

Manage Lead

Technical Data

Entity Type	Service interface
Software Component	BBPCRM
Category	Inbound
Mode	Synchronous

Definition

Group of operations to manage lead data.

Use

This inbound service interface provides basic services for creating and changing the essential lead data. It also enables you to search for leads and read their detailed data.

Integration

The inbound service interface *Manage Lead* belongs to the process component Lead Processing [Seite 104] and business object Lead [Seite 105].

Structure

The service interface groups the following inbound operations:

- LeadCreateRequestConfirmation_In (Create Lead [Seite 106])
- LeadChangeRequestConfirmation_In (Change Lead [Seite 107])
- LeadByProspectAndQualificationLevelQueryResponse_In (Read Lead [Seite 109])

Error Handling

Each service operation has a log structure that lists any errors that occur.

Create Lead

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	LeadCreateRequestConfirmation_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_LEADCRTRC

Definition

Creates a lead with its basic data.

Use

This inbound service operation enables you to create leads quickly in the normal manner in various contexts, with a selection of the data normally available, thus making it lean. During lead creation the business partner, generally a sales prospect or contact person, is assigned to the lead and the information on the lead qualification level and product interest recorded. The employee responsible specified is generally the lead qualifier, who is responsible for qualifying and processing the lead.

Features

- The lead is created for the specified business partner (for example, brand owner), with the main header data, this being the processing type code, name, classification data of group, priority, and origin, qualification level code, and period (start and end).
- Lead items include product quantity and description.
- The following standard partner functions are available for leads:
    - Sales prospect
    - Contact person
    - Employee responsible
    - Sales partner
    - Sales representative
    - Employee responsible at channel partner
- Texts with more detailed information can be assigned to the lead.
- The transaction type is prefilled and is not available for selection on the interface.
- The organizational data is determined in the background and is not available on the interface. How the organizational data is determined depends on Customizing in the back-end system. It may be the e-mail address that is used, for example.
- The lead document can be stored in an incomplete state, that is without the employee responsible assigned, and the data completed at a later point in time.
- Although it is possible to specify a role, the system determines the role based on the partner function specified, and ignores this entry, unless there is no role defined in the back-end system.
- The confirmation returns the lead ID and header details.
- The operation processes the following message types:
    - LeadCreateRequest_sync
    - LeadCreateConfirmation_sync

Change Lead

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	LeadChangeRequestConfirmation_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_LEADCHGRC

Definition

Changes a lead with its basic data.

Use

This inbound service operation enables you to change leads quickly and update their data accordingly.

As in the course of normal lead processing, it may be necessary to open and process the same lead a number of times, until processing is completed. A lead may, for example, be saved with a certain qualification level, and then reprocessed again at a later date.

The data recorded can be used to trigger follow-up processes in the back-end system, such as the creation of an opportunity when a certain qualification level is reached.

Features

- The operation changes the lead data accordingly, this being the header data (processing type code and name), the classification data (group, priority, and origin), and the qualification level code and period (start and end).
- At lead item level it changes the product quantity, product, and product description, or adds new product items.
- It changes or adds new texts.
- It changes partner function data:
  - Sales prospect
  - Contact person
  - Employee responsible
  - Sales partner
  - Sales representative
  - Employee responsible at channel partner
- The operation processes the following message types:
  - LeadChangeRequest_sync
  - LeadChangeConfirmation_sync

Read Lead

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	LeadByProspectAndQualificationLevelQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_LEADBYPROSPQUALQR

Definition

Query to and response from *Lead Processing* for retrieving one or multiple leads. For more information, see Find Lead by Prospect and Qualification Level [Seite 110].

Query Lead

Technical Data

Entity Type	Service interface
Software Component	BBPCRM
Category	Inbound
Mode	Synchronous

Definition

Service interface containing an operation to retrieve a list of leads based on different criteria.

Use

The process component *Lead Processing* does not differentiate between the search and read details function. Thus both the *Read Lead* and *Find Lead by Prospect and Qualification Level* are based on the same service operation. In both cases the data retrieved is the same, and comprises the leads found as the result of the search and their detailed data.

Integration

The inbound service interface *Query Lead* belongs to the process component Lead Processing [Seite 104] and business object Lead [Seite 105].

Structure

The service interface contains the inbound operation LeadByProspectAndQualificationLevelQueryResponse_In (Find Lead by Prospect and Qualification Level [Seite 110]).

Error Handling

Each service operation has a log structure that lists any errors that occur.

Find Lead by Prospect and Qualification Level

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	LeadByProspectAndQualificationLevelQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_LEADBYPROSPQUALQR

Definition

Retrieves lead documents based on the prospect and qualification level of the lead.

Use

This inbound service operation enables you to both search for and retrieve a full list of details about the leads concerned. It enables you to search for leads based on a particular sales prospect or with a certain qualification level or status. However, it also allows you to search flexibly using the following parameters:

- Lead ID and name
- Sales prospect, employee responsible, processing type code
- Priority and origin
- Qualification level
- Status

The result of the search is a comprehensive overview in the form of a complex list, detailing each lead found with its entire structure. The information provided comprises the item data, texts, attachments, status, and references to activities and opportunities.

Features

- The operation processes the following message types:
  - LeadByProspectAndQualificationLevelQuery_sync
  - LeadByProspectAndQualificationLevelResponse_sync

Service Confirmation Processing

Technical Data

Entity Type	Process component
Software Component	BBPCRM

Definition

Service confirmation processing is used to handle services rendered for a service order. This includes reporting back working times, materials used, as well as any expenses incurred during the service activities. These particulars are used as a basis for keeping track of working times, updating stock levels for spare parts, processing invoices, and cost accounting.

Use

You can use this business process to confirm working times, materials used, and expenses for services performed. To simplify the task of confirmation, you can plan these confirmation items in a service process (for example, a service order) or an in-house repair order. The field service representative then references the process or order and copies the relevant planned items to a service confirmation document.

Constraints

The service operations provided within this component do not work with in-house repair orders.

Notes for SAP Back-End Administration

Prerequisites

To use the operations contained in this process component, you have to be using the application components *Service Confirmation* (CRM-BTX-SCO) and *Service Processes* (CRM-BTX-SVO), and the components that are a prerequisite for working with these components.

Service Confirmation

Technical Data

Entity Type	Business object
Software Component	BBPCRM

Definition

A record of services and spare parts that a service technician reports back after performing a service for a customer. A service confirmation documents actual working times spent and spare parts used for the service. These particulars are used as a basis for processing customer invoices, updating stock levels for spare parts, carrying out cost accounting, and keeping track of working times.

Use

A service confirmation is a message from a service technician confirming the spare parts used for, time spent working on, and expenses for a service activity for a customer. It is used to document the actual working hours spent, parts used, and expenses to provide a service. It contains all information needed for further processing in the form of invoicing, material inventory management, cost center accounting, general ledger accounting, and time management.

The service confirmation is subdivided into items, which are items of a customer-specific business transaction that focuses on delivering goods or providing a service, on the prices, and on preparing the invoice. The item consists of the identifying and administrative item information for a service confirmation and contains, in addition to the schedule lines, all the data that applies to the item: in other words, the product information, the parties involved, the agreements on sales, delivery, and invoicing, the status, references, and so on.

Manage Service Confirmation

Technical Data

Entity Type	Service interface
Software Component	BBPCRM
Category	Inbound
Mode	Synchronous

Definition

Service interface containing an operation to manage service order data.

Use

This inbound service interface enables a service confirmation to be created. This can be an unplanned service confirmation, or a planned service confirmation based on a service order to which the external service provider has been assigned.

Integration

The inbound service interface *Manage Service Confirmation* belongs to the process component Service Confirmation Processing [Seite 111] and business object Service Confirmation [Seite 111].

Structure

The service interface contains the inbound service operation ServiceConfirmationForServiceProviderCreateRequestConfirmation (Create Service Confirmation for Service Provider [Seite 113]).

Error Handling

Each service operation has a log structure that lists any errors that occur.

Create Service Confirmation for Service Provider

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	ServiceConfirmationForServiceProviderCreateRequest_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_SRVCONFSRVPROVCRTRC

Definition

Creates a service confirmation for a service provider.

Use

This inbound service operation enables an external service provider to create either an unplanned service confirmation, or a planned service confirmation for a service order to which he has been assigned and which he has executed.

The service confirmation contains all data required from the service provider, describing how the service order was executed and confirming working time, used spare parts, and expenses.

Features

- The details of the services performed include the following key information:
  - Working times (duration, start and end of work)
  - Services (list of items with product ID, description, quantity, unit of measurement, net value)
  - Spare parts (list of items with product ID, description, quantity, unit of measurement, net value)
  - Expenses incurred from personnel capacities, cost of service parts, and accrued costs (list of sub-items with product ID, description, quantity, unit of measurement, amount, and currency per unit)
  - Reference object information (IBase, IBase component, object)
  - Subject codes
  - Notes
  - Executing party and date
- The service operation returns the service confirmation number.
- The operation processes the following message types:

- ServiceConfirmationForServiceProviderCreateConfirmation_sync
- ServiceConfirmationForServiceProviderCreateRequest_sync

Service Order Processing

Technical Data

Entity Type	Process component
Software Component	BBPCRM

Definition

Handles short-term agreements between a customer and a service provider, in which the customer orders one-off services. Such an order could be, for example, to maintain or repair some equipment, making it necessary to send a technician along with spare parts. These services are usually billed. The service order is also used to schedule resources and plan availability of spare parts, and can include extra expenses required to execute service jobs (for example, travel expenses).

Use

This process component provides a number of core services designed primarily for scenarios where service orders need to be accessed externally, as when working with external service providers, for example.

In cases where external service providers perform work for a particular manufacturer, the manufacturer has typically made agreements with the external service provider describing how much work is to be performed within a specific time period.

Before an external service provider confirms his work, he requires details of the arrangement to be charged during the resource assignment process.

The operations available therefore allow lists of service orders to be retrieved flexibly based on different criteria, these being essentially service provider, service assignment, and business partner.

Notes for SAP Back-End Administration

Prerequisites

To use the operations contained in this process component, you have to be using the application component *Service Processes* (CRM-BTX-SVO), and the components that are a prerequisite for this component.

Service Order

Technical Data

Entity Type	Business object
Software Component	BBPCRM

Definition

An agreement between a service provider and a customer about the execution of services at a specific time and for a specific price. In addition, the service order contains planning for personnel, spare parts, and other expenses that are necessary for providing the services.

Use

Service orders represent customer orders to service providers for the delivery of services, which can possibly include the exchange of spare parts in an after sales scenario. Service orders support service and spare parts planning and other expenses required to execute service jobs. Service order processing includes validation of existing contract and warranty entitlements, assignment of work items to the right service personnel resources and tight integration with enterprise logistics.

Manage Service Order

Technical Data

Entity Type	Service interface
Software Component	BBPCRM
Category	Inbound
Mode	Synchronous

Definition

Service interface containing an operation to manage service order data.

Use

This inbound service interface can be used to retrieve service order details for a specific service order.

Integration

The inbound service interface *Manage Service Order* belongs to the process component Service Order Processing [Seite 114] and business object Service Order [Seite 114].

Prerequisites

The service order must already exist in the back-end system.

Structure

The service interface contains the inbound operation ServiceOrderForServiceProviderByServiceOrderQueryResponse (Read Service Order For Service Provider [Seite 116]).

Error Handling

Each service operation has a log structure that lists any errors that occur.

Read Service Order For Service Provider

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	ServiceOrderForServiceProviderByServiceOrderQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_SRVORDSRVPROVBYORDIDQR

Definition

Retrieves the details of a service order for a service provider (business partner) based on the service order ID.

Use

This inbound service operation retrieves the details of a specific service order. It can therefore be used by an external service provider to obtain the details of the arrangement made concerning the work he is to undertake.

The service operation reads all attributes of the specified service order and service order items and provides this data, thus enabling the service provider to both execute and confirm the service order.

Features

- The query is executed based on the service order ID (mandatory), and optionally transaction type, service order item ID, item category, and assignment ID. If the optional parameters are specified, only those items meeting the criteria are read.
- The essential details returned include:
    - Service order header data, such as ID, description, status, priority, dates
    - Partner information, such as sold-to party, bill-to party, employee responsible
    - Reference object information (IBase, IBase component, object, product)
    - Item list with services or spare parts
    - Item details, such as assignment dates
    - Expenses from personnel capacities, cost of service parts, and accrued costs
    - Subject codes
    - Lifecycle status
    - Notes

- The operation processes the following message types:
  - ServiceOrderForServiceProviderByServiceOrderQuery_sync
  - ServiceOrderForServiceProviderByServiceOrderResponse_sync

Prerequisites

The service order ID must be known.

Query Service Order

Technical Data

Entity Type	Service interface
Software Component	BBPCRM
Category	Inbound
Mode	Synchronous

Definition

Group of operations to retrieve a list of service orders based on different criteria.

Use

This inbound service interface can be used to search for service orders by different criteria, such as service assignment or business partner.

Integration

The inbound service interface *Query Service Order* belongs to the process component Service Order Processing [Seite 114] and business object Service Order [Seite 114].

Structure

The service interface groups the following inbound operations:

- ServiceOrderSimpleByServiceAssignmentQueryResponse (Find Service Order by Service Assignment [Seite 117])
- ServiceOrderByPartyQueryResponse (Find Service Order by Party [Seite 118])

Error Handling

Each service operation has a log structure that lists any errors that occur.

Find Service Order by Service Assignment

Technical Data

Entity Type	Service operation

Technical Name in Enterprise Service Repository (ESR)	ServiceOrderSimpleByServiceAssignmentQueryResponse_In
Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_SRVORDSRVPROVBYASSIGNMQR

Definition

Provides the service order directly related to the service assignment.

Use

This inbound service operation can be used in situations where service orders have been created and directly assigned to an external service provider.

The operation searches for a service order to which a specific external service provider or service employee (resource) is assigned by service resource planning, and provides the identification of the service order for the selected service assignment.

Features

- The query is executed based on the assignment ID.
- The information returned includes the service order ID, transaction type, service order item, ID, and item category.
- The operation processes the following message types:
  - ServiceOrderSimpleByServiceAssignmentQuery_sync
  - ServiceOrderSimpleByServiceAssignmentResponse_sync

Prerequisites

The resource planning functions within the Workforce Management component are required for this service operation.

Constraints

Workforce Management is not available in SAP CRM 5.1.

Find Service Order by Party

Technical Data

Entity Type	Service operation
Technical Name in Enterprise Service Repository (ESR)	ServiceOrderByPartyQueryResponse_In

Namespace in ESR	http://sap.com/xi/CRM/SE/Global
Software Component Version in ESR	SAP CRM ABAP 5.1
Category	Inbound
Mode	Synchronous
Related Web Service Definition	CRM_SRVORDSRVPROVBYPARTYQR

Definition

Provides service orders based on the business partner.

Use

This inbound service operation can be used to find all service orders to which a specific business partner is assigned, and can therefore be used when direct assignments do not exist. The business partner may occur in different business partner roles, such as sold-to-party, bill-to party, employee responsible.

For example, an external service provider would like to see to which and how many service orders he is assigned. Though there is no direct assignment between him and the service order, he is assigned within the order with a specific partner function. Based on his business partner ID, it is therefore possible to retrieve a list of service orders.

So as not to list all service orders, additional critieria are available to filter the list as necessary:

- Date to specify, for example, service orders for the current day, week, month, or a specified time range
- Status to retrieve, for example, all service orders that are not yet completed

Features

- The query is executed based on the business partner ID, business partner function, system status, start and end dates, and reference object.
- The service order ID and description, transaction type, service order item ID, item category, sold-to party, system status, net value, priority, requested start and end dates, are returned for each service order contained in the list.
- The operation processes the following message types:
  - ServiceOrderByPartyQuery_sync
  - ServiceOrderByPartyResponse_sync