

webMethods Optimize for SAP Solutions Guide

Version 8.0 SP1

April 2016

This document applies to webMethods Optimize for SAP Version 8.0 SP1 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2007-2016 Software AG, Darmstadt, Germany and/or Software AG USA Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Table of Contents

About this Guide	7
Document Conventions.....	7
Online Information.....	8
Concepts and Architecture	9
Overview.....	10
Optimize for SAP and the SAP Process Monitoring Agent.....	10
Process, Error, and Business Data.....	11
Business Objects.....	11
Business Object Events.....	11
Receiver Function Modules.....	12
Event-Receiver Couplings.....	12
Remote Function Module WM_PUSH_EVENT.....	12
Architecture.....	13
Approach.....	13
The SAP System.....	14
Integration Server.....	16
SAP Business Process Event Mapping.....	18
Optimize for SAP Monitoring.....	19
Optimize.....	19
My webMethods Server.....	19
Sample Order to Cash Process.....	20
Steps in the Sample Order to Cash Process.....	20
Optimize for SAP Home Page.....	24
Identifying and Extending Business Objects	25
Overview.....	26
Identifying Business Objects.....	26
Browsing Business Objects.....	26
Viewing Events.....	26
Extending Business Objects.....	27
Sample Scenario.....	28
Creating Subtypes.....	28
Defining Events for Subtypes.....	29
Defining Subtypes as Delegation Types.....	30
Making the New Events Available For Use.....	30
Trigger New Events Using Message Control.....	31
ALE Message Control Settings for the Order to Cash Sample Process.....	31
Event Message Control Settings for the Order to Cash Sample Process.....	32
Defining Output Types.....	32
Defining Procedures.....	32

Defining Condition Records.....	33
Defining a New Access Sequence.....	33
Trigger New Events Using Change Documents.....	35
Adding Business Data as an Event Parameter.....	36
Modeling the Process on webMethods.....	39
Overview.....	40
Creating Adapter Notifications.....	40
Remote Event-Receiver Coupling Notification.....	40
Local Event-Receiver Coupling Notification.....	41
ALE Listener Notification.....	41
RFC Listener Notification.....	41
Creating Remote or Local Event-Receiver Coupling Adapter Notifications.....	41
Enabling Remote or Local Event-Receiver Coupling Adapter Notifications.....	44
Building the Process Model in Designer.....	44
Adapter Notifications as Receive Steps.....	45
Correlation Field or Service.....	45
Defining KPIs.....	47
Defining Order KPIs.....	47
Defining Delivery KPIs.....	49
Dealing with Latency.....	51
Configuring Optimize for SAP on SAP.....	53
Overview.....	54
Event-Receiver Couplings.....	56
Remote Event-Receiver Couplings.....	56
Local Event-Receiver Couplings.....	56
Configuring Remote Event-Receiver Couplings.....	56
Defining and Implementing a Local Correlation Function Module.....	57
Step 1: Define the Local Correlation Function Module.....	57
Step 2: Implement the Local Correlation Function Module to Alter the Event Container.....	57
Configuring Local Event-Receiver Couplings.....	60
Identifying Business Documents.....	60
Defining and Implementing a Local Receiver Function Module.....	61
Viewing the Function Interfaces for the Remote Function Module.....	62
Step 1: Define the Local Receiver Function Module.....	64
Step 2: Implement the Local Receiver Function Module to Handle Events and Push Data.....	65
Step 3: Extend the Receiver Function Module for Process Steps Not Triggered By an Event (Optional).....	66
Defining and Implementing a Local Correlation Function Module.....	67
Step 1: Define the Local Correlation Function Module.....	67
Step 2: Implement the Local Correlation Function Module to Alter the Event Container.....	68

Viewing Business Data.....	71
Viewing Optimize for SAP Business Data.....	72
Viewing KPIs for Optimize for SAP.....	72
Viewing Process Instance Details.....	74
Analyzing System Resource Monitoring Data.....	75

About this Guide

This guide describes how to use the webMethods Optimize for SAP solution (referred to as Optimize for SAP in the rest of this guide). It contains information for administrators of the webMethods platform as well as for SAP ABAP developers, process developers, and business analysts.

While this solution can be extended to address more than one SAP business process, this guide focuses on a sample Order to Cash process of the SAP Sales and Distribution (SD) Module. For more information, see "[Sample Order to Cash Process](#)" on page 20.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention <i>folder.subfolder:service</i> .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
<i>Italic</i>	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
Monospace font	Identifies text you must type or messages displayed by the system.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the symbol.
[]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [] symbols.

Convention	Description
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <http://documentation.softwareag.com>. The site requires Empower credentials. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

1 Concepts and Architecture

■ Overview	10
■ Architecture	13
■ Sample Order to Cash Process	20
■ Optimize for SAP Home Page	24

Overview

webMethods Optimize for SAP enables you to extract process data (such as the start and end time of a process instance), error data, or business data (such as customers, order quantities, and revenues) from a running SAP process. The extracted data can then be sent, using the webMethods SAP Adapter (SAP Adapter) and Integration Server technologies, to webMethods Optimize for Process (Optimize) for analysis and monitoring.

By analyzing this data, you can identify and eliminate problems and take advantage of business opportunities. For example, you can view the purchase order or invoice trend of a particular customer from a particular region and then analyze whether or not working with that customer is beneficial for your organization.

Optimize for SAP uses the monitoring capabilities of Optimize to analyze and monitor SAP business events. You indicate what events you want to monitor by identifying and extending business object events. The system then uses that information for analysis and monitoring.

As part of analysis, you can view the data as graphs, reports, and so on, using webMethods Monitor. webMethods Monitor provides you with maximum visibility and control of processes. You can use webMethods Monitor to:

- View the status of processes and steps.
- Suspend, resume, resubmit, and stop processes.
- View error and activity messages.
- Edit and resubmit processes at specific steps.

The webMethods Monitor user interface is available in My webMethods (the webMethods Monitor interface component must be installed in My webMethods Server, either at the time of initial installation, or afterward). For more information about working with webMethods Monitor, see the PDF publication *webMethods Monitor User's Guide*. For more information about working with My webMethods, see the PDF publication *Working with My webMethods*.

This chapter introduces the main concepts and describes the architecture of Optimize for SAP. It also describes, at a high level, the sample Order to Cash business process that is used as an example throughout this guide.

Optimize for SAP and the SAP Process Monitoring Agent

Optimize for SAP does not refer to a single software component. Rather, it refers to a collection of software components in combination with certain configuration settings and user-created SAP function modules.

The main software component of Optimize for SAP is the `webMethodsSAP Process Monitoring Agent`. The `SAP Process Monitoring Agent` extends the SAP Adapter by providing:

- Two additional adapter notification templates, which allow listening to business events from the SAP system (**Remote Event-Receiver Coupling (asynchronous)** and **Local Event-Receiver Coupling (asynchronous)**)
- A predefined remote function module, `WM_PUSH_EVENT`, that allows data to be pushed from the SAP system to Optimize

Process, Error, and Business Data

An SAP business process executes a series of steps to complete some business activity. Each step in the process starts, executes, and stops. The start and stop times are key elements of process data used by Optimize to monitor a process. Within SAP, a business process step is usually represented by a single Logical Unit of Work (LUW) that might either succeed or fail. Error data provided to Optimize can be quickly exposed to critical users who can identify and fix problems in the process. The intrinsic Key Performance Indicators (KPIs) in Optimize are designed to handle this data.

Some process steps can create or change the business data that the process is designed to handle. You can quickly expose this data through Optimize with custom KPIs.

Business Objects

An SAP *business object* ties together functions of a business application (business process) and the data related to those functions. A *business object type* defines a business object and its functions much like a class in object-oriented development. A *business object instance* is an instance of a business object type.

Business Object Events

A *business object event* notes a change in the state of a business object instance. When some change occurs to a business object instance within a business process, an event is fired, and an event receiver is notified. For example, if a business process application changes some item of customer data, the application sends a Changed event to the event receiver for the Customer business object, passing with it the customer number to indicate which customer's data was changed.

Optimize for SAP supports both local and remote events. Typically, events are local to the SAP system. A remote event occurs when the SAP Adapter receives an IDoc sent by the SAP system. If processed by an asynchronous ALE listener notification, the IDoc processing status can be forwarded to Optimize.

A *business object event parameter* is a parameter that is defined for a business object event and is transferred with the event if set during runtime.

Receiver Function Modules

In SAP, a *receiver function module* is a function module that allows an interested receiver to receive business object events. The interested receiver provides the receiver function module and establishes a coupling between the receiver and the events of interest.

Optimize for SAP supports two types of receiver function modules:

- Remote receiver function modules
- Local receiver function modules

The *remote receiver function module* is predefined and installed on the SAP Adapter, remote from the SAP system. This function module works independently of s or remote function module WM_PUSH_EVENT. Events on the SAP system can push intrinsic data, as well as business data defined via the event parameters, directly to the remote receiver function module on the SAP Adapter.

Important: The remote receiver function module has access to the business data defined via the event parameters only. To get access to additional business data, you must either extend your business object by defining additional event parameters and populating them during runtime or use a instead.

A *local receiver function module* is local to the SAP system. It is a user-created function module that can be coded to handle events passing intrinsic data and to pull business data from a business process. *Intrinsic data* is information about the status of a step, for example successful completion of a SAP LUW or an error event.

Local receiver function modules on the SAP system work in conjunction with the remote function module WM_PUSH_EVENT installed on the SAP Adapter (see "[Remote Function Module WM_PUSH_EVENT](#)" on page 12).

Event-Receiver Couplings

An event receiver receives events when a business object is altered. An *event-receiver coupling* ties a receiver function module to the business object event. The coupling is uniquely defined as a triple consisting of the business object type, the event name, and the receiver type.

On the Integration Server, event-receiver couplings are represented as adapter notifications. This allows events triggered by SAP to be represented as receive steps within Designer.

Remote Function Module WM_PUSH_EVENT

Remote function modules are function modules that run external to an SAP system. The SAP Process Monitoring Agent ships with one remote function module, WM_PUSH_EVENT, that is predefined on the SAP Adapter.

Local receiver function modules that you implement on the SAP system push intrinsic and business data to WM_PUSH_EVENT running on the SAP Adapter. WM_PUSH_EVENT in turn populates the pipeline with the received data, enabling you to push the data to Optimize.

WM_PUSH_EVENT is a generic container that can transport the following data transacted during the step:

- Event information
- Correlation information
- Error data
- Business data

Architecture

This section describes the approach Optimize for SAP takes to send business process data to Optimize and describes how the approach is implemented across the SAP and webMethods products.

Approach

While SAP provides several ways to send data out of a process flow, Optimize for SAP focuses on using event-receiver couplings to map a receiver function module to a business object event.

Optimize for SAP supports two different implementations of this approach. One uses remote event-receiver couplings. The other uses local event-receiver couplings.

Important: The local event-receiver coupling approach is deprecated. Functionality will not be developed further and may be removed in a later release of Optimize for SAP.

Approach	Description
Remote event-receiver coupling	Directly calls the SAP Adapter to pass intrinsic and business data to Optimize. Pushing of business data beyond the intrinsic data will only be supported as long as matching event parameters are defined with the business object event.
Local event-receiver coupling	Uses one or more local receiver function modules on the SAP system to push intrinsic and business data to remote function module WM_PUSH_EVENT on the SAP Adapter. The remote function module then populates the pipeline allowing you to forward the data to Optimize.

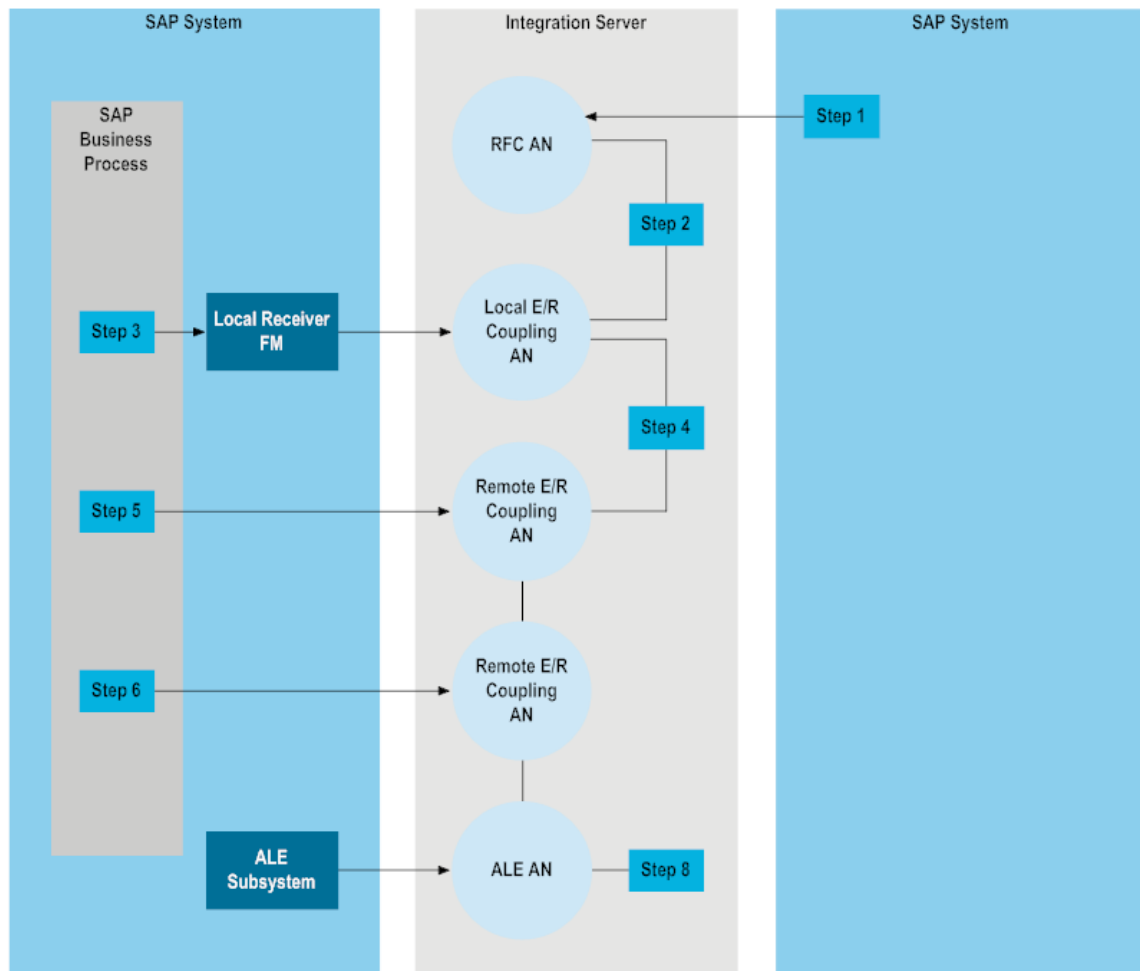
Approach	Description
	<p>A usually will invoke remote function module WM_PUSH_EVENT once for a step in a process. Each step requires its own call to WM_PUSH_EVENT, but for most steps you may not want to specify all the optional fields defined with WM_PUSH_EVENT.</p> <p>For example, in a multistep process, you would provide values for the required fields for all steps in the process. However, you would set values for the optional error fields only for those steps where a critical error occurred. And you would provide business data values for only one or two steps in the process.</p> <p>In general, you should use the remote event-receiver coupling approach over the local event-receiver coupling approach.</p>

The SAP System

The SAP system runs the complete business process or parts of the business process for which you will collect data analyzed by Optimize.

In the SAP system, the Event Type Linkage links an event to a receiver function module as represented by an adapter notification (AN) on the webMethods platform.

SAP System to Integration Server Communication



- To push intrinsic data as well as business data from SAP, you can use one of the following approaches:
 - Identify already existing business object events that have the business data you are interested in already defined as event parameters.
 - Define new business object events, including the event parameters for the demanding data, and trigger them during process execution.
 - Create a on the SAP system that hooks into the running process and invokes the WM_PUSH_EVENT remote function module (for more information about WM_PUSH_EVENT, see "[Remote Function Module WM_PUSH_EVENT](#)" on page 12).
- To push only intrinsic data from SAP, you typically identify or define new business object events.

Regardless of the data being pushed, you must define an event-receiver coupling adapter notification on the SAP Adapter for the matching business object event. In the

sample Order to Cash process, the business objects are from the Order to Cash process (as executed in SAP) of the Sales and Distribution Module. The event-receiver couplings are the hooks that push the intrinsic and business data out of the SAP business process.

- When using a remote event-receiver coupling, the SAP Adapter directly receives the business object event.
- When using a local event-receiver coupling, remote function module WM_PUSH_EVENT must be invoked from the local receiver function module. Depending on the data processed at the various steps in the process, your local receiver function module can also set additional fields ERRORTYPE, ERRORMESSAGE, and ERRORMESSAGEDETAIL or fill table BUSINESSDATA with data. The local event-receiver coupling adapter notification's signature is the same as the parameters in WM_PUSH_EVENT.

Integration Server

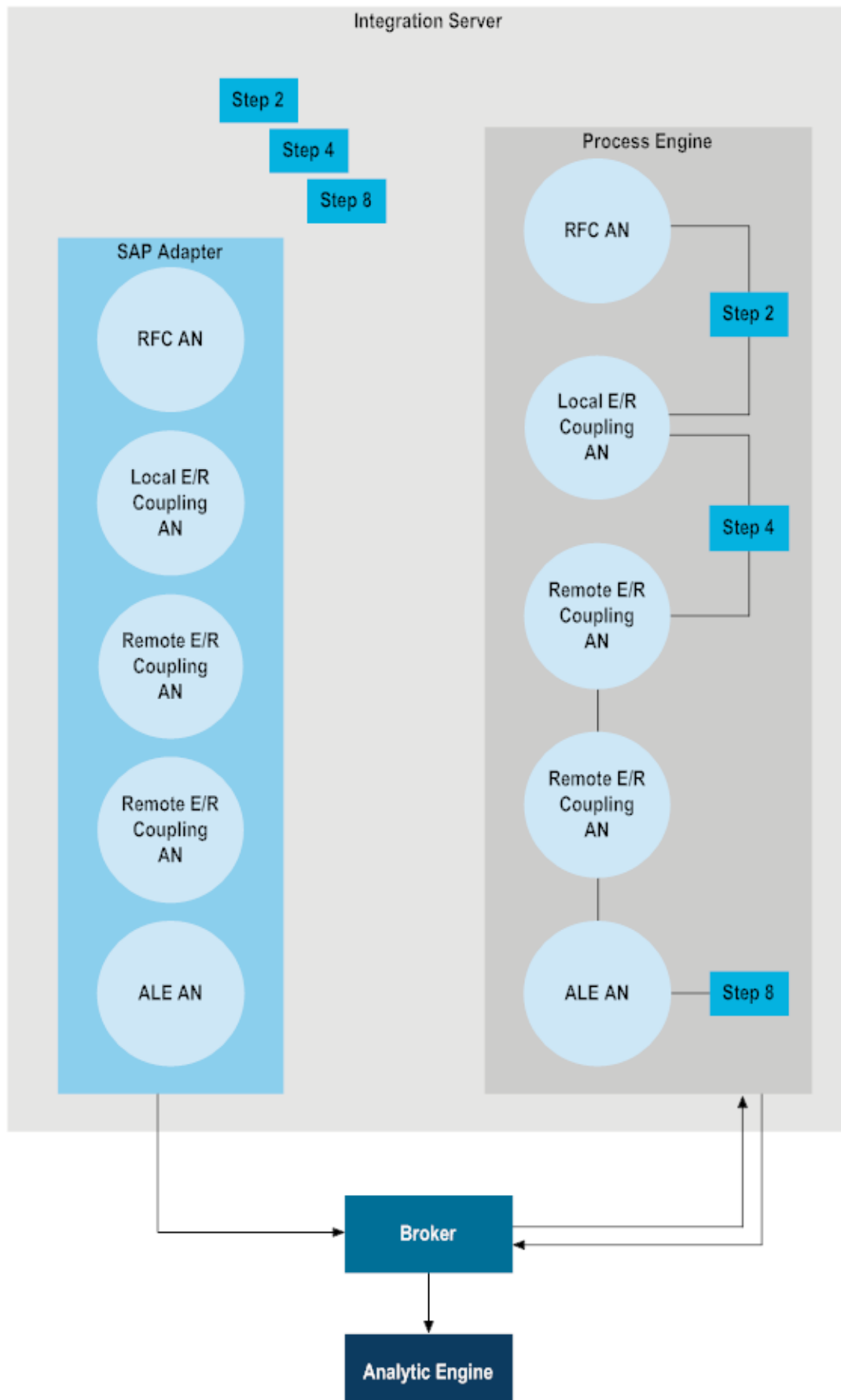
The Integration Server hosts the WmSAP package (for the SAP Adapter) and WmSAPOptimize package (for the SAP Process Monitoring Agent), as well as the Process Engine.

The WmSAP package provides the SAP Adapter runtime functionality that supports an RFC listener to receive data from SAP as well as adapter notification templates for asynchronous ALE and RFC processing. Beyond providing the Optimize for SAP runtime environment, also called the SAP Process Monitoring Agent, the WmSAPOptimize package extends the WmSAP package by providing two additional adapter notification templates for event-receiver couplings.

The business object events received from SAP as well as IDocs or plain RFC calls received from SAP will be forwarded to Universal Messaging or Broker by means of an asynchronous adapter notification.

By representing the asynchronous adapter notifications as receive steps in a business process model uploaded for execution by the Process Engine, the Process Engine will subscribe to the events received from SAP and forward them to the Analytic Engine.

Integration Server to Analytic Engine Communication



SAP Business Process Event Mapping

Optimize for SAP takes the webMethods-centric approach to SAP business processes. All events from a SAP business process are represented as asynchronous adapter notifications at the webMethods platform. The following mappings between usage patterns and the use of adapter notifications applies:

If you listen to event triggers by...	Use this adapter notification via the SAP Adapter...
Using remote event-receiver couplings	Remote Event-Receiver Coupling (asynchronous)
Using local event-receiver couplings	Local Event-Receiver Coupling (asynchronous) The publishable document created with this type of notification conforms to the import parameters defined with remote function module WM_PUSH_EVENT. This is because the Local Receiver FM specified under Receiver FM is expected to call RFC WM_PUSH_EVENT in the end.
Remotely processing IDocs	ALE Listener Notification (asynchronous)
Calling RFC WM_PUSH_EVENT directly from ABAP for process steps not triggered by an event	RFC Listener Notification (asynchronous) for function name WM_PUSH_EVENT Only one RFC Listener Notification (asynchronous) for function name WM_PUSH_EVENT needs to be created. You can assign it to different steps from your process model by specifying a subscription filter. Make sure that the RFC Listener Notification (asynchronous) created for function name WM_PUSH_EVENT is the last one in the list of Adapter Notifications defined for one listener. You can edit the notification order via Adapters > SAP Adapter > Edit Listener > Edit Notification Order .

Monitoring of business data beyond the intrinsic data is supported with all scenarios above with the restriction that remote event-receiver couplings allow you to monitor only business data that are defined as event parameters.

Optimize for SAP Monitoring

All Optimize for SAP monitoring is performed via the Process Engine. By uploading the business process for execution and letting the SAP Process Monitoring Agent communicate with the Process Engine directly, you can embed SAP business process steps into an overall business process orchestrated by the Process Engine. Additionally, the already available functionality of receive steps, like selecting a correlation service and specifying logged fields, will automatically become available in Optimize for SAP scenarios.

The Process Engine controls the run-time execution of business processes. The Process Engine is a webMethods Integration Server package that you install on every Integration Server that is used to run steps in a business process designed in Software AG Designer.

If you have distributed processes, all Integration Servers that run steps in those processes are connected to a Universal Messaging or a webMethods Broker. The Universal Messaging or the Broker is the communication link for all of the Process Engines on the connected Integration Servers. The group of Process Engines running on the Integration Servers connected to a single Universal Messaging or Broker is defined as a *Process Engine cluster*.

Optimize

Optimize provides monitoring and analysis of the business process running on SAP. Optimize includes intrinsic Key Performance Indicators (KPIs) that enable process and error-data monitoring out of the box. For business data monitoring, you can easily set up additional KPIs that are specific for your business process. For more information about configuring KPIs, see *Administering webMethods Optimize*.

By representing activities from SAP as receive steps in a business process model and uploading it for execution to the Process Engine, there is no difference in the approach when compared with any business process orchestrated by the Process Engine.

If processes are enabled for analysis, you must connect the Process Engines to the Optimize Analytic Engines so My webMethods can display the process metrics. A Universal Messaging or a Broker connects the Process Engines to the Analytic Engines.

For more information about Optimize architecture, see *Administering webMethods Optimize*.

My webMethods Server

My webMethods Server hosts My webMethods, which provides a series of user interfaces for viewing data and process diagrams.

Sample Order to Cash Process

A business process consists of many steps, and at each step some action occurs that can generate business or intrinsic data that you want to capture in Optimize. The sample Order to Cash process illustrates this processing, and is used in other sections in this guide to exemplify concepts or instructions. The provided sample process model is just one possible way to model the SAP Order to Cash business process in webMethods. Certain events could be managed differently by modeling the process differently.

The intention of the sample Order to Cash process model is not to provide a general template for Order to Cash event management but to exemplify the various event management options between the SAP system and the webMethods platform.

The sample process is a .process file that can be imported into Software AG Designer, and is available for download from the Software AG Developer Community for webMethods: <http://communities.softwareag.com/webmethods>. For instructions on importing the file, see the *Software AG Designer Online Help*.

Steps in the Sample Order to Cash Process

The following table describes all the steps in the Order to Cash process model.

Notification/Event	Represented as Receive Step...	For more information see...
Notification: IDOCORDERSinputFinished Listens to Event: IDOCORDERS.inputFinished	Electronic Order Received	"Configuring Remote Event-Receiver Couplings" on page 56
Notification: IDOCORDCHGinputFinished Listens to Event: IDOCORDCHG.inputFinished	Electronic Order Change Received	"Configuring Remote Event-Receiver Couplings" on page 56
Notification: IDOCORDERSinputErrorOccurreNotification: Listens to Event: IDOCORDERS.inputErrorOccurred	Electronic Order Create Error	"Configuring Remote Event-Receiver Couplings" on page 56

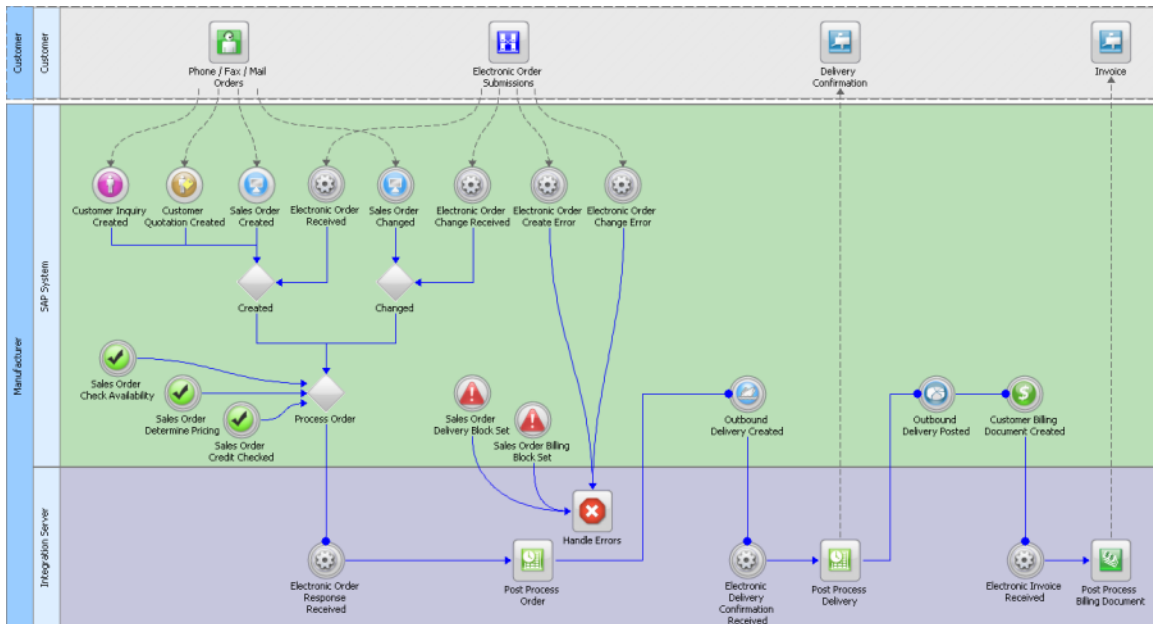
Notification/Event	Represented as Receive Step...	For more information see...
Notification: IDOCORDCHGinputErrorOccurredNotification: Listens to Event: IDOCORDCHG.inputErrorOccurred	Electronic Order Change Error	"Configuring Remote Event-Receiver Couplings" on page 56
Notification: CustomerInquiryCreatedNotification: Listens to Event: BUS2030.created	Customer Inquiry Created	"Configuring Remote Event-Receiver Couplings" on page 56
Notification: CustomerQuotationCreatedNotification: Listens to Event: BUS2031.created	Customer Quotation Created	"Configuring Remote Event-Receiver Couplings" on page 56
Notification: SalesOrderCreditCheckedNotification: Listens to Event: BUS2032.creditChecked	Sales Order Credit Checked	"Trigger New Events Using Message Control" on page 31
Notification: OutboundDeliveryPostedNotification: Listens to Event: LIKP.posted	Outbound Delivery Posted	"Trigger New Events Using Message Control" on page 31
Notification: CustomerBillingDocumentCreated Notification: Listens to Event: VBRK.created	Customer Billing Document Created	"Trigger New Events Using Message Control" on page 31
Notification:	Sales Order	"Trigger New Events Using

Notification/Event	Represented as Receive Step...	For more information see...
<p>SalesOrderDeliveryBlockSetNotification: Listens to Event: BUS2032.deliveryBlockSet</p>	<p>Delivery Block Set</p>	<p>Message Control" on page 31</p>
<p>Notification: SalesOrderBillingBlockSetNotification: Listens to Event: BUS2032.billingBlockSet</p>	<p>Sales Order Billing Block Set</p>	<p>"Trigger New Events Using Change Documents" on page 35</p>
<p>Notification: OutboundDeliveryCreatedNotification: Listens to Event: LIKP.created</p>	<p>Outbound Delivery Created</p>	<p>"Adding Business Data as an Event Parameter" on page 36</p>
<p>Notification: SalesOrderCreatedNotification: Listens to Event: BUS2032.created</p>	<p>Sales Order Created</p>	<p>"Configuring Local Event-Receiver Couplings" on page 60</p>
<p>Notification: SalesOrderChangedNotification: Listens to Event: BUS2032.changed</p>	<p>Sales Order Changed</p>	<p>"Configuring Local Event-Receiver Couplings" on page 60</p>
<p>Notification: SalesOrderDeterminePricingNotification: Listens to Event: BUS2032.determinePricing</p>	<p>Sales Order Determine Pricing</p>	<p>"Step 3: Extend the Receiver Function Module for Process Steps Not Triggered By an Event (Optional)" on page 66</p>

Notification/Event	Represented as Receive Step...	For more information see...
Notification: WM_PUSH_EVENTNotification: Listens to Event: n/a	Sales Order Check Availability	"Step 3: Extend the Receiver Function Module for Process Steps Not Triggered By an Event (Optional)" on page 66
Notification: IDOCORDRSPNotification: Listens to Event: n/a	Electronic Order Response Received	<i>webMethods SAP Adapter Installation and User's Guide</i>
Notification: IDOCDESADVNotification: Listens to Event: n/a	Electronic Delivery Confirmation Received	<i>webMethods SAP Adapter Installation and User's Guide</i>
Notification: IDOCINVOICNotification: Listens to Event: n/a	Electronic Invoice Received	<i>webMethods SAP Adapter Installation and User's Guide</i>

The following figure illustrates the sample Order to Cash process as modeled in Software AG Designer:

Sample Order to Cash Process Flow



Optimize for SAP Home Page

The webMethods Optimize for SAP home page of your Integration Server installation contains a summary of information about the product as well as sample ABAP code for various tasks that Optimize for SAP can perform. To access the home page, go to **Integration Server > Solutions > Optimize for SAP**.

2 Identifying and Extending Business Objects

■ Overview	26
■ Identifying Business Objects	26
■ Extending Business Objects	27
■ Sample Scenario	28

Overview

This chapter describes how to determine what business objects are relevant to your process and if they have defined events that are being triggered during runtime. Also, it explains how to create new events and ensure that they get triggered during runtime if events for what you want to monitor do not exist.

Identifying Business Objects

Before you can create the process model, you need to determine what business objects are relevant to your business process.

Browsing Business Objects

Within your SAP system, you can use TCode SWO3 to browse the Business Object Repository (BOR).

For example, for the sample Order to Cash process:

- Go to **Application components > Sales and distribution > Sales**. To display the relevant business objects:
 - CustomerInquiry (data model BUS2030)
 - CustomerQuotation (BUS2031)
 - SalesOrder (BUS2032)
 - OutboundDelivery (LIKP)
- Go to **Application components > Sales and distribution > Billing** to find the additional business object VBRK.

Viewing Events

For business objects you have found, you can expand the tree labeled **Events**.

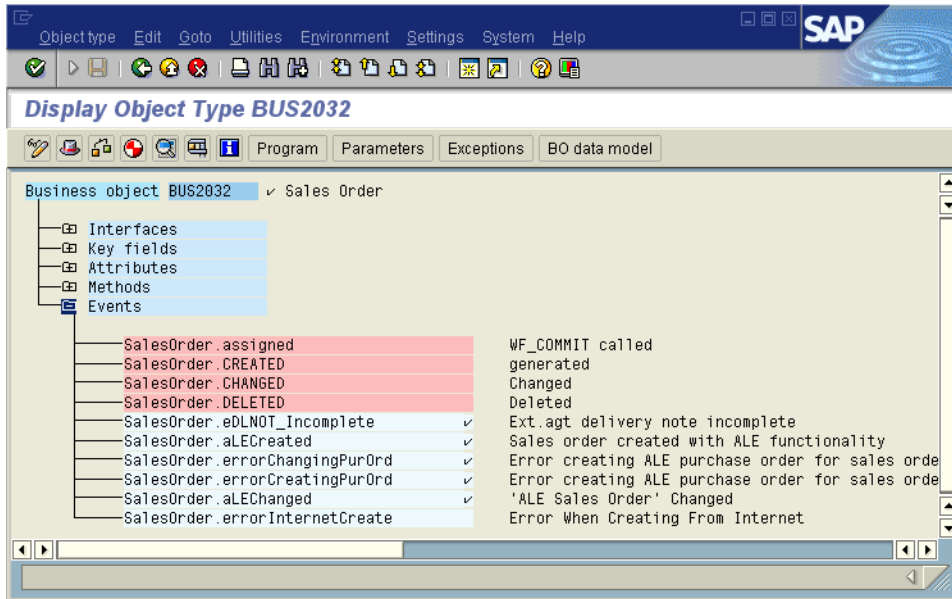
Note: To identify the business object events related to a business process, you can enable event tracing in SAP using TCode SWELS. With event tracing enabled, execute a typical instance of the process. You will be able to monitor the business object events that are already defined and triggered by the process.

Also, when you browse the BOR, notice that CustomerInquiry, CustomerQuotation, and SalesOrder are derived from the same business object called SalesDocument (VBAK). This business object has the following predefined events:

- SalesDocument.CREATED

■ SalesDocument.CHANGED

Sample business object events



Extending Business Objects

If no event for what you want to monitor exists, you can create a new event using the SAP Business Object Builder. When using the Business Object Builder to create custom events, you create event names that are defined against the business object type.

Note: The event trigger code will be part of the ABAP program that runs the specific business process. SAP User-Exits and Business Add-Ins can provide an appropriate way to add event creation code to existing SAP programs without modifying them.

Tip: As an alternative to coding your own event trigger code, you can use generic mechanisms such as Change Documents, Status Management, Message Control, Logistic Information System (LIS), and Business Transaction Events. For more information about these features, see the SAP documentation library.

Your SAP standard system comes with pre-defined business object events used by the Sample Order to Cash process. To provide even more visibility into the business process, the following additional events are included:

Business Object Event	Description
BUS2032.creditChecked	Sales Order Credit Checked

Business Object Event	Description
BUS2032.deliveryBlockSet	Sales Order Delivery Block Set
BUS2032.billingBlockSet	Sales Order Billing Block Set
BUS2032.determinePricing	Sales Order Determine Pricing
LIKP.posted	Outbound Delivery Posted
VBRK.created	Customer Billing Document Created

Sample Scenario

The procedures below illustrate the steps necessary for extending business objects.

Creating Subtypes

Before continuing, you must define a new package in SAP where all the newly created objects can be stored. Use TCode SE80 to define the new package, and name it something that is easy to identify, such as Z_WEBM_EVENT.

Instead of defining new events with the existing business object types, you can create a new subtype that extends an existing supertype.

To create a new subtype

1. Using TCode SWO1, in the **Object/Interface type** field, enter the supertype.
2. Click **Subtype**.
3. Save the new business object type, specifying the package attribute as the previously created package Z_WEBM_EVENT.
4. Repeat this procedure for all the business object types you want to extend by providing the following information:

Supertype	Subtype	Object Name	Name	Program	Application
BUS2032	ZBUS2032	MySalesOrder	My Sales Order	ZRBUS2032	Z

Supertype	Subtype	Object Name	Name	Program	Application
LIKP	ZLIKP	MyOutboundDelivery	My Outbound Delivery	ZSDLIKP01	Z
VBRK	ZVBRK	MyBillingDocument	My Billing Document	ZSDVBRK01	Z

Defining Events for Subtypes

To define events for newly created subtypes

1. Using TCode SWO1, enter the newly created subtype in the **Object/Interface type** field.
2. Click **Change**.
3. Position the cursor at the **Events** node and click **Create**.
4. Enter the information provided in the following table for each new event:

Object Type	Event	Name	Description
ZBUS2032	creditChecked	Credit Checked	Credit check was executed, document OK.
ZBUS2032	deliveryBlockSet	Delivery Block Set	Document blocked for delivery reasons.
ZBUS2032	billingBlockSet	Billing Block Set	Document blocked for billing reasons.
ZBUS2032	determinePricing	Determine Pricing	Pricing was determined, document OK.
ZLIKP	posted	Posted	Goods issue posting.

Object Type	Event	Name	Description
ZVBRK	created	Created	Invoice created.

5. Click **Continue**.
6. Click **Save**.

Defining Subtypes as Delegation Types

To ensure the newly defined events become accessible to the existing business logic, each of the three new subtypes must be defined as delegation types for their supertypes.

To define a new subtype as a delegation type for its supertype

Using TCode SWO6 in the edit mode:

- If the delegation was already defined for your supertype:
 1. Select the row that lists your supertype.
 2. In the **Delegation type** field, you may choose to change the value to the newly created subtype. Doing so disables your previous subtype because you can have only one delegation type for each supertype. If you would like both subtypes, either merge them or further delegate from one subtype to the other one.
 3. Click **Save**.
- If the delegation has not been defined previously:
 1. Create a new entry for the supertype.
 2. In the **Delegation type** field, change the value to the newly created subtype.
 3. Click **Save**.

Making the New Events Available For Use

So far the new events for the given object types are only listed as available. You now need to make them available for use.

To make the new events available for use

1. Using TCode SWO1, specify the subtype in the **Object/Interface type** field.
2. Click **Change**.
3. Position the cursor at the event node and select **Edit > Change release status > Object type component > To implemented**.
4. Position the cursor at the object type node and select **Edit > Change release status > Object type > To implemented**.

5. Select **Object type > Generate**.
6. The steps above described how to define additional events and make them available for use. However, at this time there are no event triggers in place. The following sections present two different generic frameworks provided by SAP that let you enable event triggers.

Trigger New Events Using Message Control

This section describes the message control settings done at your SAP system to trigger the newly created events from the Order to Cash process. Because message control is used in the context of ALE processing as well, we will also give a brief overview of the typical ALE-related message control settings.

ALE Message Control Settings for the Order to Cash Sample Process

The Sample Order to Cash process uses message types ORDRSP, DESADV, and INVOIC to electronically notify the customer about the current state of the business process. Processing of these messages will be enabled per SalesOrg/Customer. The following table lists the message control settings used by the Order to Cash sample to trigger ALE messages:

Message	ORDRSP	DESADV	INVOIC
Application	V1	V2	V3
Output/ Condition Types	BA00	LALE	RD00
Procedures	V10000	V10000	V10000
Requirements	2		4
Condition Records			
■ Partner Function	SP	SP	BP
■ Medium	A	A	A
■ Date/Time	4	4	4

Message	ORDRSP	DESADV	INVOIC
Access Sequences	0003	0005	0004

Event Message Control Settings for the Order to Cash Sample Process

The same framework used to trigger the ALE processing can be used to trigger business object events.

The new events `BUS2032.creditChecked`, `BUS2032.deliveryBlockSet`, `LIKP.posted`, and `VBRK.created` have been defined previously. The same process is used to set up message controls to trigger each of these events, but the values used differ for each one. We will use `BUS2032.creditChecked` as an example.

Defining Output Types

To define an output type

1. Using TCode NACE, select application **V1**.
2. Click **Output types**.
3. Switch to edit mode, then click **New entries**.
4. In the **Output type** field, enter `Z200`.
5. Describe the output type by entering `Credit Check`.
6. Select the following options: **Access to conditions**, **Multiple issuing**, and **Partner-independent output**.
7. Under default values, set the **Dispatch time** to **4 (Send immediately)** and **Transmission medium** to **9 (Events)**.
8. Click **Save**.
9. With application **V1** and output type **Z200** still selected, click **Processing routines** in the structure tree on the left side of the screen.
10. For **Transmission medium**, select **Events**.
11. For **Program**, enter `RSWEMC01`.
12. For **Form routine**, enter `CREATE_EVENT` as the first processing routine.
13. Click **Save**.
14. Click **Back**.

Defining Procedures

To define a procedure

1. Using TCode NACE, select application **V1**.

2. Click **Procedures**.
3. Select procedure **V10000**.
4. From the structure tree on the left of the screen, click **Control**.
5. Switch to edit mode, then click **New entries**.
6. In the **Step** field, enter a step number that has not yet been assigned.
7. Under **Condition type**, enter **Z200**, which is the output type you just created. Leave the **Requirement** field empty.
8. Click **Save**.
9. Click **Back**.
10. Using TCode **NACS**, enter **0008** as the **Access sequence** to the **Condition type** **Z200**.
11. Click **Save**.

Defining Condition Records

To define the condition records

1. Using TCode **NACE**, select application **V1**.
2. Click **Condition records**.
3. Select Output type **Z200**.
4. In the **Overall credit status** field, specify value **A**, then press **Enter**.
5. In the **Medium** field, specify **9** and in the **Time** field specify **4**.
6. Select the newly added row.
7. In the **Goto** menu, click **Communication method**.
8. In the **Object type** field, select **ZBUS2032** and in the **Event** field, select **creditChecked**.
9. Click **Save**.
10. Click **Back**.

Defining a New Access Sequence

If there is no matching access sequence configured with your SAP standard system, you can define your own. In the following procedure, **BUS2032.deliveryBlockSet** is used as an example.

To define a new access sequence

1. Using TCode **NACQ**, in the **Application** field, enter **V1** and in the **Table** field, enter **502**.
2. In the **Condition** menu, click **Create**.
3. From the field catalog on the right side of the screen, double click **Delivery block**. The field is copied to the list of selected fields on the left side of the screen.

4. Click **Generate** to generate the condition table.
5. Click **Back**.
6. Using TCode NACE, select application **V1**.
7. Click **Access sequence**.
8. Switch to edit mode, then click **New entries**.
9. In the **Access sequence** field, enter z502.
10. Describe the output type by entering `Delivery block`.
11. Select the row that you just added.
12. From the structure tree on the left of the screen, click **Access**.
13. Enter an access number that has not yet been assigned.
14. In the **Condition table** field, enter 502.
15. Click **Save**.
16. Click **Back**.

This procedure can be repeated for all new events using the data provided in the following table:

Event	ZBUS2032. creditChecked	ZBUS2032. deliveryBlockSet	ZLIKP.posted	ZVBRK. created
Application	v1	v1	v2	v3
Output/ Condition Types	z200	z502	z013	z013
Procedures	v10000	v10000	v10000	v10000
Requirement			1	4
Condition Records				
■ Overall Credit Status	A			
■ Delivery Block		01...09		

■ Sales Organization ^a			XXXX	XXXX
■ Medium	9	9	9	9
■ Date/Time	4	4	4	4
Access Sequences	0008	Z502	0013	Z013
Condition Table	200	502	013	013

a. Enter your 4-digit sales organization code here.

Note: All condition tables with a value of less than 500 already exist in the SAP standard system. Access sequences starting with a 'Z' are usually not yet defined in the SAP standard system and you will need to create them using TCode NACE.

Trigger New Events Using Change Documents

The SAP standard system comes with a change document object for sales documents called VERKBELEG.

To assign an event to a change document

1. Using TCode SWEC, switch to edit mode.
2. Click **New entries**.
3. In the **Change document object** field, enter VERKBELEG.
4. For **Object category**, select **BOR Object type**.
5. In the **Object type** field, enter ZBUS2032.
6. In the **Event** field, enter `billingBlockSet`.
7. For **Trigger event**, select **On change**.
8. Click **Save**.
9. With the new entry still selected, choose **Field restrictions** from the structure tree on the left side of the screen.
10. Open the **Condition editor**.
11. In the combined field restriction table, add one new line with **Expression 1** set to `&VBAK_FAKSK_NEW&` and the **Operator** set to **Not equal**.

12. Click **Back**.
13. Click **Save**.
14. From now on, every time the **Billing block (VBAK_FAKSK)** field from your sales order is changed to a non-null value, the event `ZBUS2032.billingBlockSet` will be triggered.

Adding Business Data as an Event Parameter

In addition to tracking a business process and transferring intrinsic data to the webMethods platform, you may want to include business data in the events triggered by the SAP system. Certain events defined in the SAP Business Object Repository are already defined with a rich set of parameters. However, by default, other events will transmit only the intrinsic data. In this section, using an existing `OutboundDelivery.created` event, we show how you can extend a business object event to transfer additional business data.

To allow a business object event to transfer additional business data

1. Using TCode SWO1, in the **Object/Interface type** field, enter the previously created subtype `ZLIKP`. Click **Change**.
2. Position the cursor at the **MyOutboundDelivery.created** node and in the **Edit** menu, click **Redefine**.
3. With the cursor still positioned at **MyOutboundDelivery.created**, click **Parameters**.
4. Click **Create**.
5. When asked if you want to **Create with ABAP dictionary field proposal**, click **No**.
6. Provide the following information in the opened window:

In this field...	Enter this...
Parameter	<code>createdSets</code>
Name	<code>Created Sets</code>
Description	<code>Created Sets</code>
Multiline	<code>Yes</code>
Reference Table	<code>ENT6033</code>

7. Click **Continue**.
8. Click **Save**.
9. Click **Back**.

10. Position the cursor at **MyOuboundDelivery.created** again and select **Edit > Change release status > Object type component > To released**.
11. Select **Object type > Generate**.

Note: Using TCode SWO6, make sure the altered subtype ZLIKP with the newly created events is defined as a Delegation type for supertype LIKP.

12. For now you have only defined the new parameter for the event LIKP.created. To ensure that the event container received by the webMethods platform will also contain the parameter list **createdSets** that corresponds to the SAP entity ENT6033, you must ensure that the entity ENT6033 has been added to the event container. You would usually do this as part of the local correlation function module. For more information see "[Defining and Implementing a Local Correlation Function Module](#)" on page 57.

3 Modeling the Process on webMethods

■ Overview	40
■ Creating Adapter Notifications	40
■ Building the Process Model in Designer	44

Overview

Before using Optimize for Process to monitor and analyze an SAP process, you must first model that process using Software AG Designer (Designer). These tasks include the following:

1. Configure adapter notifications for the events related to the process using webMethods Developer.
2. Build a model of the business process, associating adapter notifications with receive steps in the model using Software AG Designer.
3. Deploy the process model to Integration Server using Software AG Designer.

Creating Adapter Notifications

For each step in your process triggered by an event (local or remote), create an adapter notification to allow mapping of events to receive steps (of the business process). There are three types of notification templates provided in the SAP Adapter: event-receiver coupling notification (local and remote), ALE notification, and RFC notification. The sample Order to Cash process uses all three types.

Remote Event-Receiver Coupling Notification

The Remote Event-Receiver Coupling notification template is used for the following steps:

- Electronic Order Received
- Electronic Order Change Received
- Electronic Order Create Error
- Electronic Order Change Error
- Customer Inquiry Created
- Customer Quotation Created
- Sales Order Credit Checked
- Sales Order Determine Pricing
- Sales Order Delivery Block Set
- Sales Order Billing Block Set
- Outbound Delivery Created
- Outbound Delivery Posted

- Customer Billing Document Created

Local Event-Receiver Coupling Notification

The Local Event-Receiver Coupling notification template is used for the following steps:

- Sales Order Created
- Sales Order Changed

ALE Listener Notification

The ALE Listener Notification template is used for the following steps:

- Electronic Order Response Received
- Electronic Delivery Confirmation Received
- Electronic Invoice Received

For complete information about ALE notifications, see the *webMethods SAP Adapter Installation and User's Guide*.

RFC Listener Notification

The RFC Listener Notification template is used for the following step:

- Sales Order Check Availability

For complete information about RFC notifications, see the *webMethods SAP Adapter Installation and User's Guide*.

Creating Remote or Local Event-Receiver Coupling Adapter Notifications

Before testing an event-receiver coupling, you must create and enable a matching adapter notification.

For example, in the sample Order to Cash process, the steps in "[Remote Event-Receiver Coupling Notification](#)" on page 40 are represented by event-receiver coupling adapter notifications (remote or local).

To create a remote or local event-receiver coupling adapter notification

1. In webMethods Developer, select **File > New**.
2. Select **Adapter Notification** from the list of elements and click **Next**.
3. Select **SAP Adapter** as the adapter type and click **Next**.

4. Select the template appropriate for your operation (either **Remote Event-Receiver Coupling (asynchronous)** or **Local Event-Receiver Coupling (asynchronous)**) from the template and click **Next**.
5. Select the appropriate listener from **Notification Listener Name** and click **Next**.
Select the listener that corresponds to the default RFC destination (WMBEM) defined in the listener that was configured during installation of the SAP Process Monitoring Agent.
6. Type a unique name for the asynchronous listener notification, and select the appropriate folder. Click **Next**.
7. Click **Finish**.

The adapter notification template creates the following items:

- A remote or local event-receiver coupling notification
- A Publishable Document Type



In the adapter notification service editor, you can select the **Adapter Settings** tab at any time to confirm the following listener notification properties:

- Adapter name
- Adapter listener name
- Adapter notification template

8. Select the **Event-Receiver Coupling** tab to verify or modify the following properties:

Property	Description
Business Object Pattern	All or part of the name of the business object on the SAP system. Allows you to restrict the selection of values for property Business Object Type . Enter a wildcard-like pattern for the business object for which you want to create the notification. You can use exact patterns, or patterns with single or multiple wildcard characters.
Business Object Type	The resulting business object type, as defined in SAP, that matches the provided Business Object Pattern .
Business Object Name	The resulting business object name, as defined in SAP, that matches the provided Business Object Pattern .
Event Name	The event name for the selected business object type, as defined in SAP.

Property	Description
Correlation FM	<p>If you implemented a correlation function module for your event-receiver coupling, specify the function module name here. This function module correlates key field values from different business objects to a common one.</p> <ul style="list-style-type: none"> ■ For a remote event-receiver coupling adapter notification, this property allows you to manipulate the event container before passing it to the SAP Adapter. For information about creating a local correlation function module for use with remote event-receiver couplings, see "Defining and Implementing a Local Correlation Function Module" on page 57. ■ For a local event-receiver coupling adapter notification, this property allows you to manipulate the event container before passing it to the local receiver function module. For more information about creating a local correlation function module for use with local event-receiver couplings, see "Defining and Implementing a Local Correlation Function Module" on page 67. <p>With respect to the sample Order to Cash process, the value specified for the Correlation FM property would be Z_WM_CORRELATE_DOCUMENTS.</p>
Receiver FM	<p>(Local event-receiver coupling adapter notification only.) Specify the name of the local receiver function module that should handle events from the process step.</p> <p>With respect to the sample Order to Cash process, the value specified for the Receiver FM property would be Z_WM_HANDLE_EVENT.</p> <p>For more information about local event-receiver couplings, see "Defining and Implementing a Local Receiver Function Module" on page 61.</p>

9. Select the **Request Field Selection** tab to specify which fields should match the arriving message to run the notification.
 - Select the fields by selecting the appropriate boxes in the **Use** column.
 - To select all fields click the Check All Rows icon .
 - To de-select all fields click the Uncheck All Rows icon .


10. Select the **Permissions** tab to manage the access control list (ACL) information. Use the drop-down menu to select each of the ACL types. For general information about assigning and managing ACLs, see *webMethods Service Development Help*.
11. From the **File** menu, select **Save** (or **Save All**).
12. Enable the local event-receiver coupling, as described in the next step.

Enabling Remote or Local Event-Receiver Coupling Adapter Notifications

To enable a remote or local event-receiver coupling adapter notification

You must enable the notification for each process step that you plan to monitor via a local or remote event-receiver coupling.

1. Start Integration Server Administrator if it is not already running.
For information about starting the Integration Server Administrator, see *webMethods Integration Server Administrator's Guide*.
2. In the **Adapters** menu in the Integration Server Administrator navigation area, click **SAP Adapter**.
3. Click **Listener Notifications**.
4. On the Listener Notifications screen, click **No** in the **Enabled** column for the listener notification you want to enable.

The Integration Server Administrator enables the listener notification and displays a  and **Yes** in the **Enabled** column.

Building the Process Model in Designer

After the business analyst understands the tasks involved in a business process, the process can be modeled. The business analyst prepares the model in the Business Analyst perspective of Software AG Designer. Using Designer, the business analyst can add an activity step for each task orchestrated by the Process Engine, and can add receive steps for previously created adapter notifications, show the flow by drawing lines (or *transitions*) between the steps, and use *swimlanes* to identify the department or area that is responsible for performing each step.

After the business analyst models the flow of the business process, the model is passed to the technical staff, who add technical detail to the model. The technical staff uses the Process Developer perspective of Designer to build up the process model to make it executable.

To execute a business process, the run-time objects for the process model must exist on the run-time servers, and the process model must be enabled. Then the Process Engine will use these run-time objects when events occur that trigger the business process.

Detailed instructions on using Designer are beyond the scope of this guide. For information on using Designer to model a process, and build and upload a process model for execution, see the *Software AG Designer Online Help*. For information about enabling the process model, see *webMethods Monitor User's Guide*.

Adapter Notifications as Receive Steps

Previously created adapter notifications (see "[Creating Adapter Notifications](#)" on page 40), will now be represented as receive steps in the business process model.

In Designer, using Package Navigator, browse to the adapter notification (not the associated document type) and drag-and-drop it on the canvas. Designer will automatically select the matching Receive Document and Receive Protocol as defined by the adapter notification:

Adapter Notification Type	Adapter Notification Settings	Receive Protocol
Asynchronous	Publish Document to webMethods Broker	Subscription (For Broker Documents)
Asynchronous	Publish Document to JMS Provider	JMS (For JMS Triggered Processes)
Synchronous	Execution Mode - Publish and Wait	Simple Service (For synchronous Reply)
Synchronous	Execution Mode - Service Invoke	N/A

Note: Synchronous adapter notifications are only supported when configured to use execution mode Publish and Wait. In these scenarios, an additional reply step needs to be placed on the canvas that has **Properties > Advanced > Implementation > Reply Document** set to the adapter notifications reply document. Also, property **Properties > Advanced > Implementation > Reply To** needs to be set to the receive step that subscribes to the adapter notifications request document.

For details about the specific receive step properties, see the *Software AG Designer Online Help*.

Correlation Field or Service

When modeling a business process that consists of several receive steps, which is common in a typical Optimize for SAP environment, correlation considerations become a major part in the process development. When there is more than one receive step there

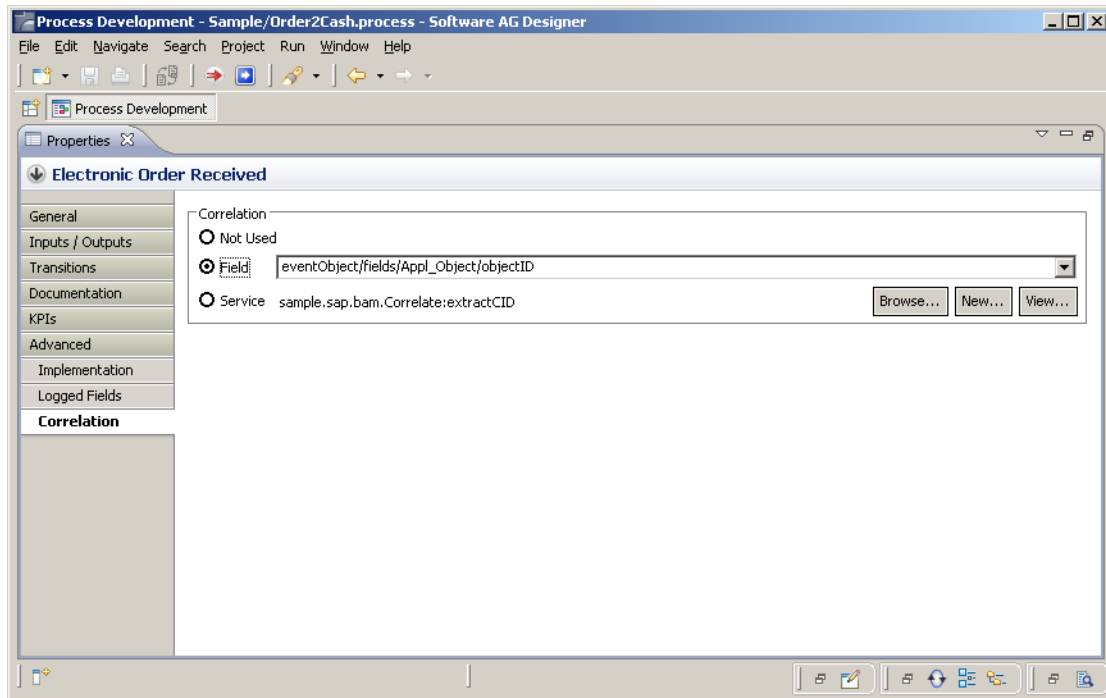
is a need to notify the Process Engine to establish, lookup, and delete a correlation ID to correctly match the received subscription document with a particular process instance.

To select a field from the subscription document as a correlation key

1. If the receive step's subscription document contains a suitable correlation key, in the receive step's **Properties > Advanced > Correlation** panel, in the **Field** list, select the field to be used as the correlation key, as shown in the figure below.

For example, in the sample Order to Cash process model, the subscription documents for receive steps Electronic Order Received and Electronic Order Change Received contain field `eventObject/fields/Appl_Object/objectID` referencing the sales order number. This field therefore qualifies as a correlation key that links receive step Electronic Order Received to receive step Sales Order Created and receive step Electronic Order Change Received to receive step Sales Order Changed.

Figure 5. Selection of a field as a correlation key



In the previous example, there is no need to make any alterations to the SAP system because event parameter `IDOCORDERS.inputFinished.Appl_Object` is already defined in the SAP standard system. In other cases, additional event parameters have to be defined (see ["Adding Business Data as an Event Parameter" on page 36](#)) before they can be selected as a correlation key. For example, for receive step Outbound Delivery Created, field `eventObject/fields/createdSets[0]/VGBEL` qualifies as a correlation key because it references the sales order number used as the correlation key in a previous receive step.

2. If the receive step's subscription document does not contain a suitable correlation key, in the receive step's **Properties > Advanced > Correlation** panel, specify a service as the Correlation Service and implement the correlation logic there. For example,

you could introspect the pipeline and concatenate certain fields to a new field. This approach provides flexibility, including call backs to other components to resolve the correlation.

The WmSAPOptimize package comes with a sample correlation service `sample.sap.bem.Correlate:extractCID` that extracts all the document numbers from the different documents involved in the Order to Cash process and forwards them to the process engine as correlation IDs. This approach lets you correlate any received subscription document to a process instance as long as it contains a reference to a document number from a previously processed document in SAP. This approach works in conjunction with the implementation of a local correlation function module. See ["Defining and Implementing a Local Correlation Function Module" on page 57](#) for details.

For more details on how to establish, look up, and delete the correlation between a correlation id and process id, see *Administering webMethods Process Engine*.



Defining KPIs

You can define KPIs directly in your process model and deploy them to the Analytic Engine by building and uploading the process model for execution.

It is beyond the scope of this document to provide a rich set of business data KPIs; however, as an example we describe a minimal set of order and delivery KPIs that might be of interest in an Order to Cash scenario.

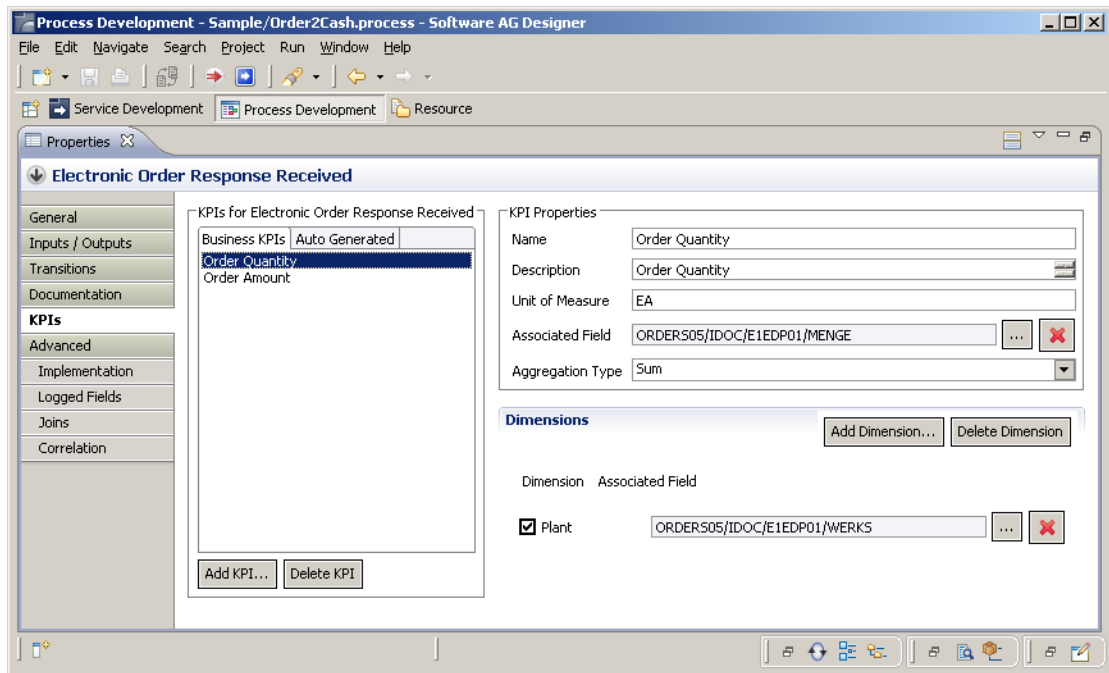
Defining Order KPIs

To define order KPIs

1. In Software AG Designer, using receive step `Electronic Order Response Received`, go to **Properties > KPIs**.
2. To define the KPI `Order Quantity`:
 - a. Click **Add KPI**, enter `Order Quantity`, and then click **OK**.
 - b. In the **KPI Properties** panel, enter a description and unit of measure for the new KPI.
 - c. Click the  icon to the right of **Associated Field** and select `ORDERS05/IDOC/E1EDP01/MENGE` from the tree.
 - d. In the **Aggregation Type** list, select **Sum**.
 - e. In the **Dimensions** panel, click **Add Dimension**, enter `Plant`, then click **OK**.
 - f. Click the  icon to the right of the new dimension and select `ORDERS05/IDOC/E1EDP01/WERKS` from the tree.

The completed screen will look like this:

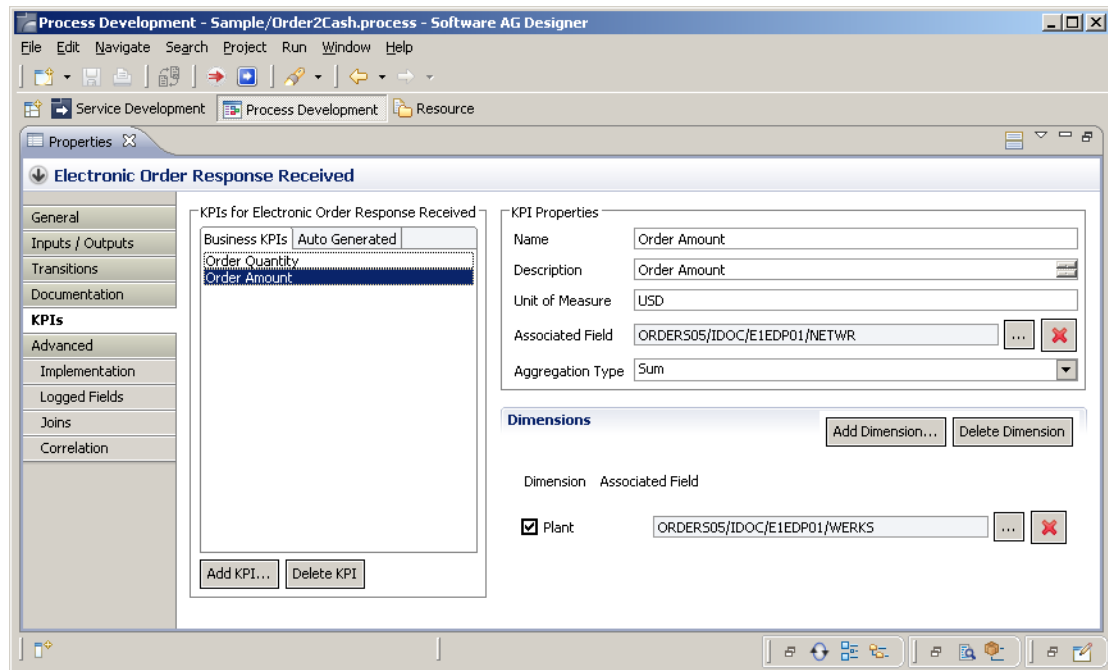
Figure 6. Example KPI Order Quantity



3. To define KPI Order Amount:
 - a. Click **Add KPI**, enter `Order Amount`, then click **OK**.
 - b. In the **KPI Properties** panel, enter a description and unit of measure for the new KPI.
 - c. Click the icon to the right of **Associated Field** and select **ORDERS05/IDOC/E1EDP01/NETWR** from the tree.
 - d. In the **Aggregation Type** list, select **Sum**.
 - e. Click the icon to the right of **Plant** and select **ORDERS05/IDOC/E1EDP01/WERKS** from the tree.




The completed screen will look like this:

Figure 7. Example KPI Order Amount



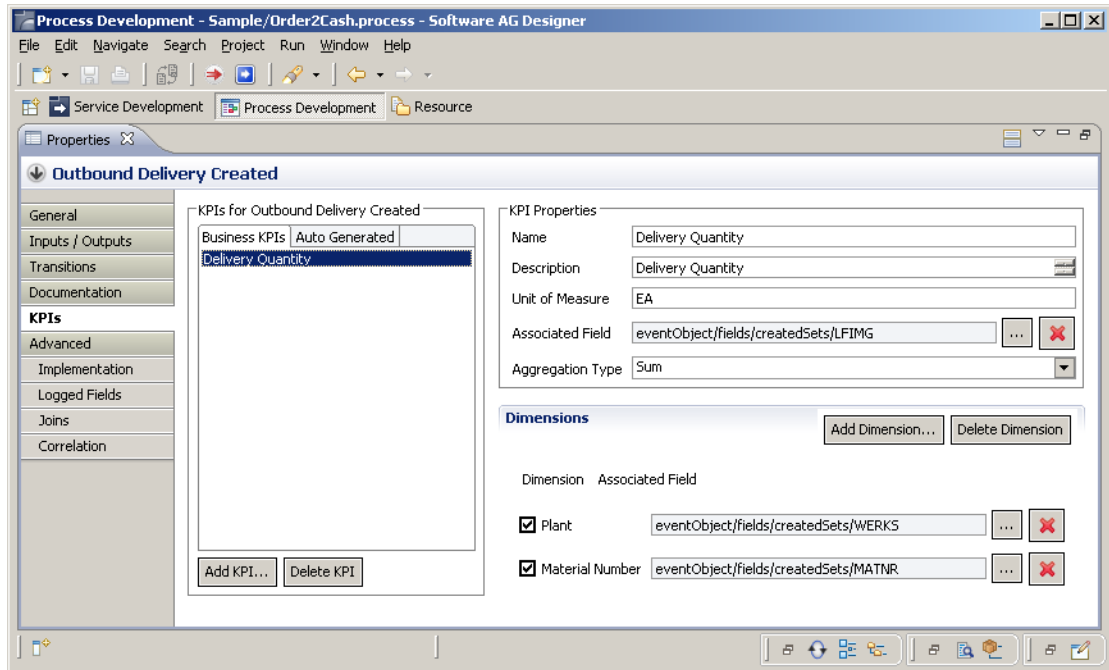
Defining Delivery KPIs

To define delivery KPIs

1. In Software AG Designer, using receive step Outbound Delivery Created, go to **Properties > KPIs**.
2. To define the KPI Delivery Quantity:
 - a. Click **Add KPI**, enter `Delivery Quantity`, then click **OK**.
 - b. In the **KPI Properties** panel, enter a description and a unit of measure for the new KPI.
 - c. Click the  icon to the right of **Associated Field** and select `eventObject/fields/createdSets/LFIMG` from the tree.
 - d. In the **Aggregation Type** list, select **Sum**.
 - e. In the **Dimensions** panel, click the  icon to the right of **Plant** and select `eventObject/fields/createdSets/WERKS` from the tree.
 - f. In the **Dimensions** panel, click **Add Dimension**, enter `Material Number`, then click **OK**.
 - g. In the **Dimensions** panel, click the  icon to the right of **Material Number** and select `eventObject/fields/createdSets/MATNR` from the tree.

The completed screen will look like this:

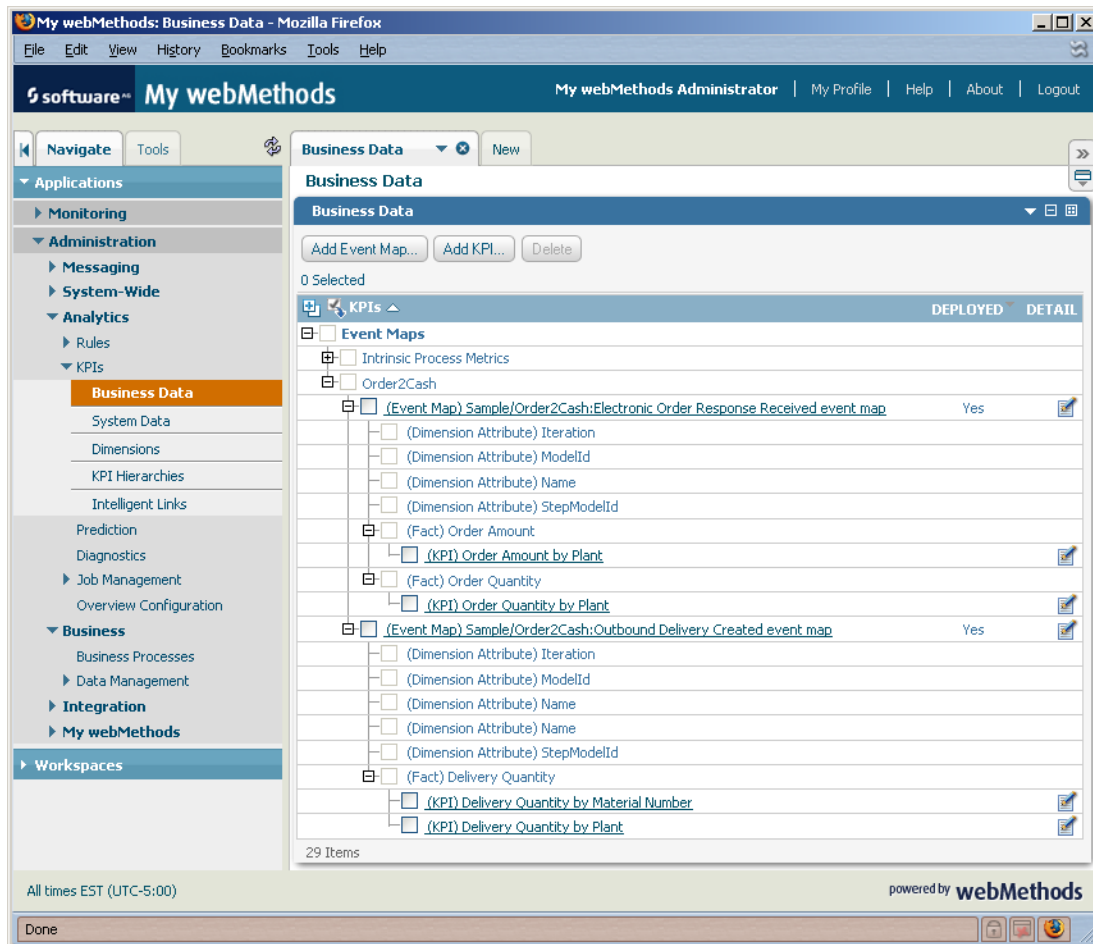
Figure 8. Example KPI Delivery Quantity



Note: For the delivery KPI, the previously defined dimension, Plant, which was used by the order KPIs, will be reused by assigning a different field from the subscription document.

3. After building and uploading your business process model for execution, use My webMethods to verify that the KPIs were defined and deployed correctly:

Figure 9. KPI Definitions



For information about the steps of specifying the event and the event map information, see *Administering webMethods Optimize*.

Dealing with Latency

After you finish modeling, you have to build and upload the process model for execution. This will create a new package at the Integration Server using the name you specified in Designer under **Properties > Advanced > Runtime > Generated Package Name**. Specifying Order2CashPE here will result in the creation of a new package Order2CashPE at the Integration Server. This package will contain the service stubs representing the steps from the business process. In addition, there will be one transition trigger per process model version and one subscription trigger per package.

The subscription trigger will have an entry for all the subscription documents. Each entry references the document type associated with the Adapter Notification that represents a receive step.

The subscription trigger property **Transient error handling > Max retry attempts** has a default value of 0. This might be a problem in scenarios where event latency plays a role. For example, with respect to the sample Order to Cash process, there is no guarantee

that event Sales Order Created will always be received before event Electronic Order Received as modeled. You have to ensure that the Broker resubmits an event if it is received out of order. This can be achieved easily by changing property **Transient error handling > Max retry attempts** to a value greater than 0.

4 Configuring Optimize for SAP on SAP

■ Overview	54
■ Event-Receiver Couplings	56
■ Configuring Remote Event-Receiver Couplings	56
■ Configuring Local Event-Receiver Couplings	60

Overview

In the previous chapter, an SAP business process was modeled in Software AG Designer (Designer). Business object events defined in the SAP Business Object Repository, as well as RFC or ALE calls to the SAP Adapter, were represented as receive steps in the model by means of adapter notifications.

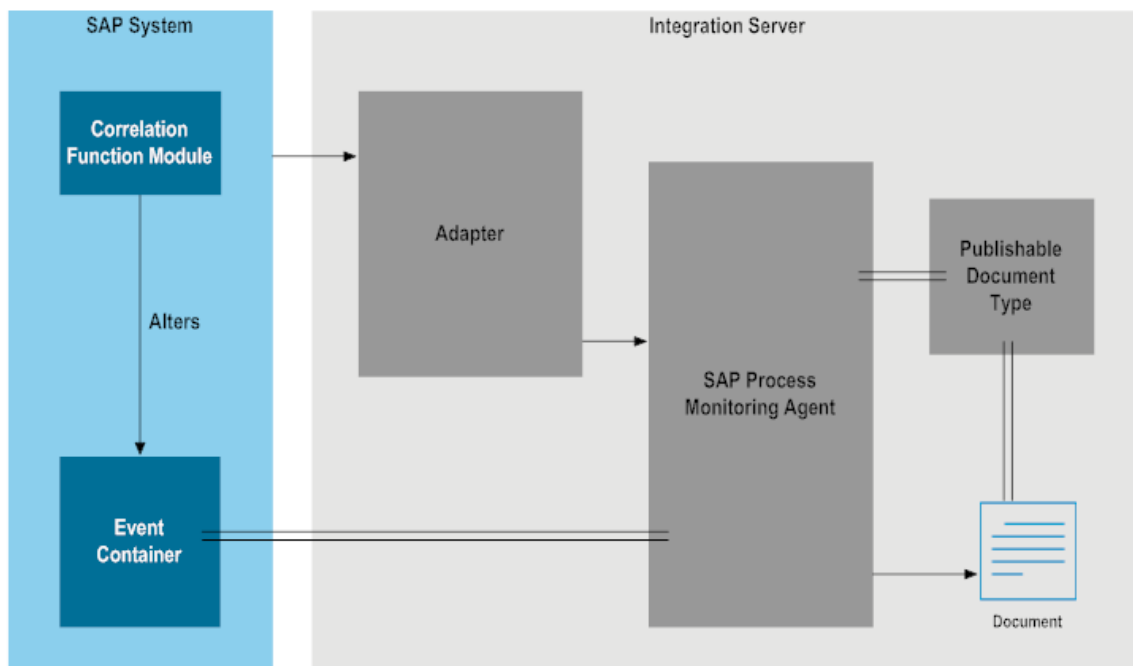
This chapter covers the tasks that must be performed on SAP to enable the SAP Process Monitoring Agent to work with the SAP business process. The following must be done:

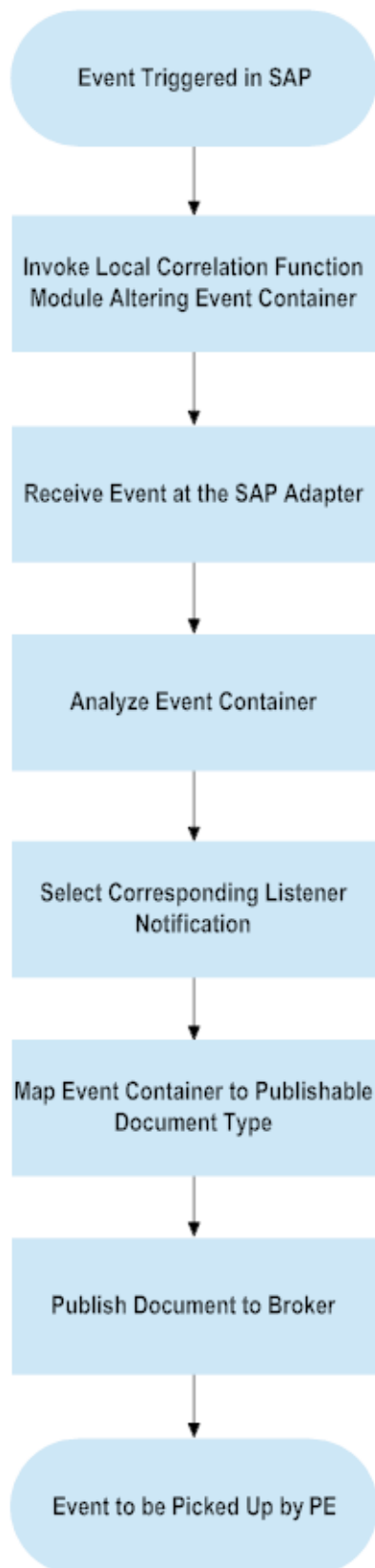
- Implement correlation function modules when dealing with more than one business object type in SAP.
- Implement local receiver function modules for local event-receiver couplings.

The SAP Process Monitoring Agent will then be able to transform the event container created by the SAP system to a publishable document type that can be managed by the Process Engine. Assigning a local correlation function module to the event-receiver coupling provides the capability to alter the event container before passing it to the webMethods platform.

The following block and flow diagrams illustrate the data flow between the SAP system and the webMethods ESB:

SAP to ESB Runtime Block Diagram



SAP to ESB Runtime Data Flow

Event-Receiver Couplings

There are two approaches to push business events from SAP to webMethods Optimize. One approach uses remote event-receiver couplings, the other uses local event-receiver couplings.

Remote Event-Receiver Couplings

With remote event-receiver couplings, event triggers push business object instance data directly to the SAP Adapter.

For the sample Order to Cash process introduced earlier in this guide (see "[Sample Order to Cash Process](#)" on page 20), you would use remote event-receiver couplings for the steps listed in "[Modeling the Process on webMethods](#)" on page 39 "Remote Event-Receiver Coupling Notification" on page 2.

For instructions on using remote event-receiver couplings, see "[Configuring Remote Event-Receiver Couplings](#)" on page 56.

Local Event-Receiver Couplings

Important: The local event-receiver coupling approach is deprecated. Functionality won't be developed further and may be removed in a later release of Optimize for SAP.

With local event-receiver couplings, event triggers push intrinsic data to a local receiver function module at the SAP system. The receiver function module uses the business object instance data to pull related business data. The receiver function module then pushes the business object instance and business data to remote function module WM_PUSH_EVENT at the SAP Adapter. For more information, see "[Approach](#)" on page 13.

For the sample Order to Cash process introduced earlier in this guide (see "[Sample Order to Cash Process](#)" on page 20), you would use local event-receiver couplings for the steps listed in "[Local Event-Receiver Coupling Notification](#)" on page 41.

For instructions on using local event-receiver couplings, see "[Configuring Local Event-Receiver Couplings](#)" on page 60.

Configuring Remote Event-Receiver Couplings

Working with remote event-receiver couplings reduces the implementation effort significantly. Instead of retrieving, mapping, and forwarding your business object instance data using ABAP, the SAP Process Monitoring Agent retrieves the data directly, transforms it, and forwards the data to Optimize automatically.

This section describes the tasks you must perform on your SAP system to use remote event-receiver couplings to extract data from business process steps and push that data to Optimize.

To use a remote event-receiver coupling, you must write local correlation function module code, as described in detail in the section that follows.

Defining and Implementing a Local Correlation Function Module

Key fields from a business object will be used as the correlation ID by the Process Engine. If you deal with several business objects, their different key field values must be correlated to one common process instance ID generated by the Process Engine.

Step 1: Define the Local Correlation Function Module

To correlate either predefined or custom events, you must define one or more local correlation function modules in your SAP system. You can define a local correlation function module to alter the event container of a specific event, many different events, or all the events you will use to push data to Optimize. You can define as many local correlation function modules as you want to alter the event container.

Any local correlation function module you define must implement the signature, as shown below:

```

*-----
*"*Local Interface:
*  IMPORTING
*    VALUE (EVENT)           LIKE      SWETYPECOU-EVENT
*    VALUE (RECTYPE)        LIKE      SWETYPECOU-RECTYPE
*    VALUE (OBJTYPE)        LIKE      SWETYPECOU-OBJTYPE
*    VALUE (OBJKEY)         LIKE      SWEINSTCOU-OBJKEY
*    VALUE (EXCEPTIONS_ALLOWED) LIKE    SWEFLAGS-EXC_OK DEFAULT SPACE
*  EXPORTING
*    VALUE (REC_ID)         LIKE      SWELOG-RECID
*  TABLES
*    EVENT_CONTAINER       STRUCTURE SWCONT
*  EXCEPTIONS
*    TEMP_ERROR
*    ANY_ERROR
*-----

```

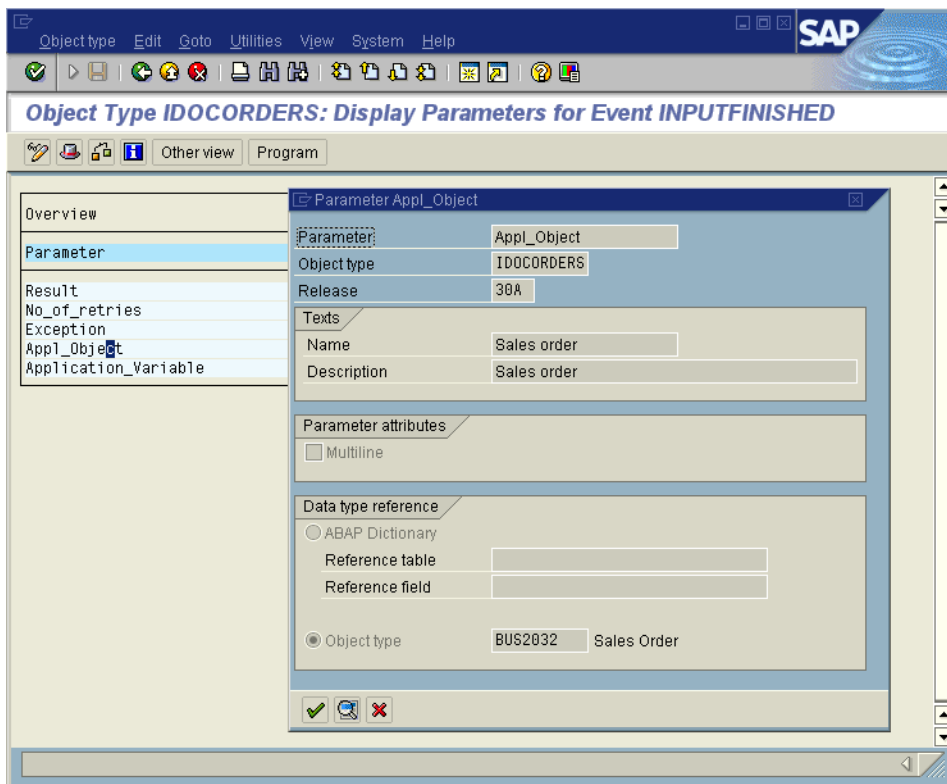
The local receiver function module's signature should conform to function module SWE_TEMPLATE_CHECK_FB that is included in the SAP standard system installation. You can create a local receiver function module by creating a copy of SWE_TEMPLATE_CHECK_FB.

Step 2: Implement the Local Correlation Function Module to Alter the Event Container

During run time, event data is delivered to the correlation function module you defined in the previous step. The data is passed in a standard event container that includes the following elements:

- Event object type
- Event name
- Event object instance
- Event object key
- Event initiator (that is, the person under whose user ID the event was created)
- Event creation date/time

If an event has additional parameters defined with it, the event container will also contain these elements. For example, the business object event IDOCORDERS.inputFinished has parameter **Appl_Object** defined as an additional parameter.



The **Appl_Object** element points to the business object instance to be created in SAP. In this case, the event parameter IDOCORDERS.inputFinished.Appl_Object corresponds with business object SalesOrder (BUS2032). Therefore, events IDOCORDERS.inputFinished and BUS2032.created can be correlated easily because event IDOCORDERS.inputFinished already comes with a reference to business object type BUS2032. The same applies for events IDOCORDCHG.inputFinished and BUS2032.changed.

If an **Appl_Object** element exists in the event container, in your correlation service registered with the Process Engine, you would use the key field value from the application object instead of the key field value of the event object. That means for an

IDOCORDERS.inputFinished event, the value from key field SalesOrder.SalesDocument will be used as document id over key field value IDOCORDERS.IDocNumber.

Beyond the additional event parameters available in the event container as described above, you can always add more fields to the event container. To later determine field **ProcessCorrelationID** in your correlation service registered with the Process Engine, you would retrieve a previously processed business object instance from the document flow and add it to the event container as field **CORR_OBJECT**. The SAP Process Monitoring Agent will check for this field and map it to the **correlationObject** field from the notification signature.

For the sample Order to Cash process, the process could be started by the Customer Inquiry Created, Customer Quotation Created, or Sales Order Created step:

- If the process-start step is Customer Inquiry Created, the inquiry number should be used as the value for the correlation ID for the next step. You could also use the inquiry number for the correlation ID for all subsequent steps in one process instance.
- If the process-start step is Customer Quotation Created, the quotation number should be used as the value for the correlation ID for the next step. You could also use the quotation number for the correlation ID for all subsequent steps in one process instance.
- If the process-start step is Sales Order Created, the order number should be used as the value for the correlation ID for all subsequent steps in one process instance.

You achieve this behavior by specifying a correlation function module in the event-receiver coupling that simply adds field **CORR_OBJECT** to the event container.

Generally speaking, when remembering all previously used document numbers as correlation ids for one process instance id at the Process Engine, it is sufficient to add the business object instance from any previously processed document from the same document flow as the value for **CORR_OBJECT** to the event container.

In your correlation function module you would call a method that determines the previously processed document from the document flow and then updates the event container with the retrieved business object instance.

If you defined additional parameters for an event, you can use the correlation function module to update the event container with the parameter values. See ["Adding Business Data as an Event Parameter" on page 36](#) for information on how to define additional parameters for an event.

```
FUNCTION z_wm_correlate_documents.
*"-----
  DATA hcl TYPE REF TO zcl_bem.

*- Initialize helper class
  CREATE OBJECT hcl
    EXPORTING
      evt_container = event_container[]
      rectype       = rectype.

*- Set event parameters
  IF objtype = zbem_likp AND event = zbem_created.
```

```

CALL METHOD hcl->set_evt_parameters( ).
ENDIF.

*- Add entry CORR_OBJECT to the container
CALL METHOD hcl->set_corr_object( ).
*- Retrieve the altered container
event_container[] = hcl->get_evt_container( ).
ENDFUNCTION.

```

Important: Before testing or running a remote event-receiver coupling, you must link the correlation function module and an event type by creating and enabling a matching adapter notification. For complete instructions, see "[Creating Remote or Local Event-Receiver Coupling Adapter Notifications](#)" on page 41. You will specify the name of your local correlation function module in the **Correlation FM** property on the **Event-Receiver Coupling** tab on the adapter notification page.

Implementing the local event-receiver coupling also requires implementing a helper class and setting global data declarations. For more information about helper class and data declarations, see the sample code for the Order to Cash process accessible via the Optimize for SAP home page of your Integration Server installation. For more information about the Optimize for SAP home page, see "[Optimize for SAP Home Page](#)" on page 24.

Configuring Local Event-Receiver Couplings

This section describes the tasks you must perform on your SAP system to use local event-receiver couplings to extract data from business process steps and push the data to Optimize.

To use a local event-receiver coupling, you must:

- Identify business documents (for extracting business data)
- Define, implement, and optionally extend, local receiver function module code
- Implement local correlation function module code

Each of these tasks is discussed in more detail in the following sections.

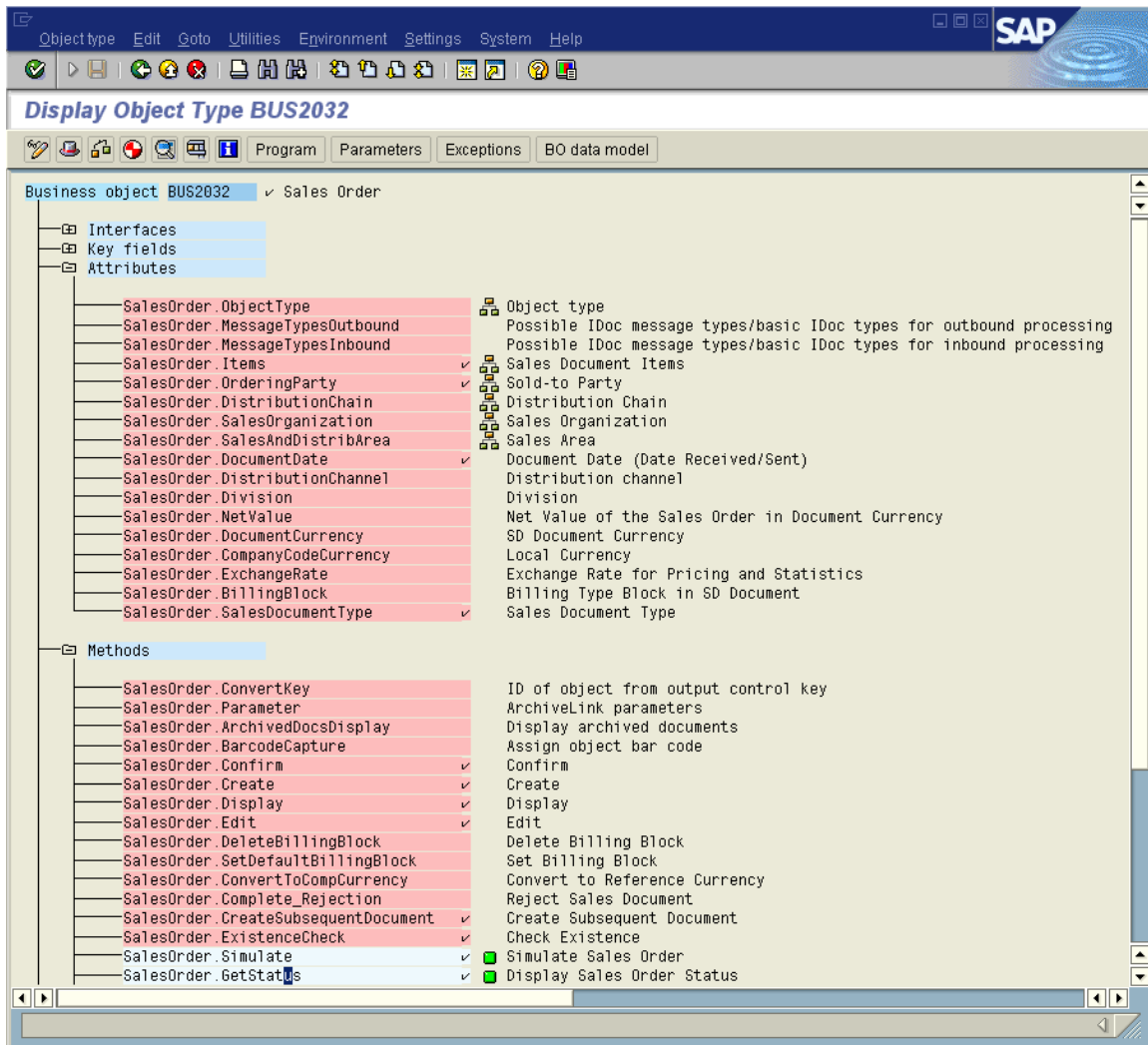
Identifying Business Documents

While event triggers push business object instance data to the local receiver function module, the local receiver function module pulls business data out of the business process. To do this, the local receiver function module needs to receive the document ID of the business document that contains the business data.

In the Business Object Repository, expand the business object tree at **Key fields**. For example, you can see that the key field for the SalesDocument business object is SalesDocument.SalesDocument (reference table VBAK, reference field VBELN).

Expand the business object tree at **Attributes**. This shows additional business data available for your business object, for example, SalesDocument.DocumentDate or SalesDocument.NetValue. Item business data is available via the referenced SalesItem business object, for example, SalesDocument.SalesItem.TargetQuantity.

Sample business object attributes and methods



To access additional business data, you also could call BAPIs (Business Application Programming Interfaces). Expanding the business object tree at **Methods** displays the simple methods and BAPIs defined with the business object. For example, calling BAPI SalesOrder.GetStatus from business object type BUS2032 allows access to the Statusinfo parameter describing the current status of a sales order.

Defining and Implementing a Local Receiver Function Module

The following sections describe the tasks you use to define and implement a local receiver function module.

Viewing the Function Interfaces for the Remote Function Module

Before you begin to implement a local receiver function module, you should examine the function interface of the remote function module WM_PUSH_EVENT.

If you implement a local receiver function module at the SAP system, it must call the remote function module WM_PUSH_EVENT defined on the SAP Adapter (see ["Concepts and Architecture" on page 9](#) "Remote Function Module WM_PUSH_EVENT" on page 6). When calling remote function module WM_PUSH_EVENT, the CALL FUNCTION statement you write for the needs to conform to the metadata for the remote function module. You will need to view this data when you implement the CALL FUNCTION statement.

To view a function interface

1. Start Integration Server Administrator if it is not already running.

For information about starting the Integration Server Administrator, see *webMethods Integration Server Administrator's Guide*.

2. In the **Adapters** menu in the Integration Server Administrator navigation area, click **SAP Adapter**.
3. Click **DDIC-Cache**.

The SAP Repository Cache table appears, showing all of the element types stored in the cache.

4. For the System ID "(local)", click the number in the **Functions** column.

Note: If the DDIC-Cache screen does not show any metadata for the System ID "(local)," you might perform a quick lookup for function module WM_PUSH_EVENT from the Lookup screen for any enabled SAP system id.

The Cached Functions table appears. To view more details about a function module:

- a. Click the name of the function module.

The function interface of the function module appears.

- b. If there is a link in the **Table** column, you can click it to see the structure definition of a parameter.

The function interface for the function module WM_PUSH_EVENT is:

Figure 13. Signatures of Function Module WM_PUSH_EVENT

Function interface for WM_PUSH_EVENT						
Class	Parameter	Table	Type	Length	Decimals	Optional
I	EVENTNAME		CHAR	32	0	No
I	EVENTOBJECTTYPE		CHAR	10	0	No
I	EVENTOBJECTID		CHAR	70	0	No
I	RECEIVERTYPE		CHAR	14	0	No
I	CORRELATIONOBJECTID		CHAR	70	0	Yes
I	TIMESTAMP_START		CHAR	15	0	Yes
I	TIMESTAMP_STOP		CHAR	15	0	Yes
I	BUSINESSDATA	WMKEYVALUE	TABLE	12	0	Yes
I	ERRORTYPE		CHAR	255	0	Yes
I	ERRORMESSAGE		CHAR	1024	0	Yes
I	ERRORMESSAGEDETAIL		STRING	8	0	Yes

The parameters are described in the following table.

Field	Description
EVENTNAME	The name of the business object event as defined in SAP. For example: CREATED
EVENTOBJECTTYPE	The name of the business object type as defined in SAP. For example: BUS2032
EVENTOBJECTID	The key field value identifying this business object instance.
RECEIVERTYPE	The receiver type received from the event container.
CORRELATIONOBJECTID	A correlation ID known to, or in need of getting registered with, the Process Engine from the same process instance.
TIMESTAMP_START	The start time stamp for the executed step in the process.
TIMESTAMP_STOP	The stop time stamp for the executed step in the process.
BUSINESSDATA	Not a scalar value. It is a table of the type WMKEYVALUE, which holds name/value pairs:

Field	Description												
Structure definition for WMKEYVALUE													
	<table border="1"> <thead> <tr> <th>Field</th> <th>Length</th> <th>Decimals</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>NAME</td> <td>64</td> <td>0</td> <td>CHAR</td> </tr> <tr> <td>VALUE</td> <td>255</td> <td>0</td> <td>CHAR</td> </tr> </tbody> </table>	Field	Length	Decimals	Type	NAME	64	0	CHAR	VALUE	255	0	CHAR
Field	Length	Decimals	Type										
NAME	64	0	CHAR										
VALUE	255	0	CHAR										
	<p>The table contains a list of name/value pairs that holds the actual metric data, as collected by the local receiver function module. In the case of an Order to Cash process, typical metric data points include ItemCount, OrderChangeCount, ItemQuantity, ItemAmount, ProductID, and others. Optimize uses this metric data as KPIs to monitor and analyze the SAP business process.</p> <ul style="list-style-type: none"> ■ NAME identifies a single entity of business data. ■ VALUE contains the actual value of this entity. 												
ERRORTYPE	Indicates the type of error that occurred.												
ERRORMESSAGE	A shortened version of the error message.												
ERRORMESSAGEDETAIL	The full version of the error message.												

Step 1: Define the Local Receiver Function Module

To handle either predefined or custom events, you must define one or more local receiver function modules in your SAP system. You can define a local receiver function module to handle a specific event, many different events, or all the events you will use to push data to Optimize. You can define as many local receiver function modules as you want to handle your implementation.

Any local receiver function module you define must implement the signature, as shown below:

```

*-----
*""Local interface:
*  IMPORTING
*    VALUE (EVENT)           LIKE      SWETYPESCOU-EVENT
*    VALUE (RECTYPE)        LIKE      SWETYPESCOU-RECTYPE
*    VALUE (OBJTYPE)        LIKE      SWETYPESCOU-OBJTYPE
*    VALUE (OBJKEY)         LIKE      SWEINSTCOU-OBJKEY
*    VALUE (EXCEPTIONS_ALLOWED) LIKE    SWEFLAGS-EXC_OK DEFAULT SPACE
*  EXPORTING
*    VALUE (REC_ID)         LIKE      SWELOG-RECID
*  TABLES
*    EVENT_CONTAINER       STRUCTURE SWCONT
*  EXCEPTIONS
*    TEMP_ERROR

```



```

*"      ANY_ERROR
*"-----

```

The local receiver function module's signature should conform to the function module `SWE_TEMPLATE_REC_FB` that is included in the SAP standard system installation. You can create a local receiver function module by creating a copy of `SWE_TEMPLATE_REC_FB`.

Step 2: Implement the Local Receiver Function Module to Handle Events and Push Data

During run time, event data are delivered to the local receiver function module you defined in the previous step. You must map this data to remote function module `WM_PUSH_EVENT`, defined at the SAP Adapter. The data are passed in a standard event container that includes the following elements:

- Event object type
- Event name
- Event object instance
- Event object key
- Event initiator (that is, the person under whose user ID the event was created)
- Event creation date/time

To easily extract the relevant data from the event container, include the following library into your helper class used by the receiver function module:

```
INCLUDE cntn01_swc      " container macros
```

For example, starting with the basic `Z_WM_HANDLE_EVENT` function module for the Order to Cash process, the following sample shows how to implement the previously defined local receiver function module to map the event data from the SAP system to remote function module `WM_PUSH_EVENT` on the SAP Adapter. This enables the receiver function module to push the data.

The basic sequence of steps for implementing a local receiver function module are:

1. Extract the business object key field value from the received event container.
2. Optionally, retrieve the business object instance for this key field value.
3. Push business object instance and business data to Optimize using remote function module `WM_PUSH_EVENT`, defined in the SAP Adapter.

Sample local receiver function module `Z_WM_HANDLE_EVENT`

```

FUNCTION z_wm_handle_event.
*"-----
  DATA hcl TYPE REF TO zcl_bem.

  *- Initialize helper class
  CREATE OBJECT hcl
  EXPORTING
    evt_container = event_container[]
    rectype       = rectype.

```

```

*- Set business data
IF objtype = zbem_bus2032 AND event = zbem_created.
  CALL METHOD hcl->set_business_data( ).
ENDIF.

*- Push the data to webM
CALL METHOD hcl->push_event( ).

*- Handle substeps
IF objtype = zbem_bus2032.
  CALL METHOD hcl->handle_substeps( ).
ENDIF.
ENDFUNCTION.

```

As shown above, local receiver function module Z_WM_HANDLE_EVENT pushes business data for the following step from the business process model: Sales Order Created.

To push data for the following steps, you might extend the Z_WM_HANDLE_EVENT function module: Sales Order Determine Pricing, Sales Order Check Availability. For more information on extending the function module, see ["Step 3: Extend the Receiver Function Module for Process Steps Not Triggered By an Event \(Optional\)"](#) on page 66.

Implementing the local event-receiver coupling also requires implementing a helper class and setting global data declarations. For more information about helper class and data declarations, see the sample code for the Order to Cash process accessible via the Optimize for SAP home page of your Integration Server installation.

Step 3: Extend the Receiver Function Module for Process Steps Not Triggered By an Event (Optional)

Some of the steps in your business process might not match exactly with an event definition in your SAP system. However, you can trigger additional events or invoke an RFC adapter notification directly from within your local receiver function module implementation as substeps, or subroutines, to one event-linkable step. You would do so by querying the business documents and then evaluating the related data identified in ["Identifying Business Documents"](#) on page 60.

The sample Order to Cash process comes with the following two steps that show how an event can be created from any program by calling the relevant function module:

- Sales Order Check Availability: Invokes an RFC adapter notification directly by calling remote FM WM_PUSH_EVENT.
- Sales Order Determine Pricing: Triggers an additional event by calling FM SWE_EVENT_CREATE (SAP_WAPI_CREATE_EVENT) that will then be handled by a remote event-receiver coupling.

For more information, see the sample code for the Order to Cash process accessible via the Optimize for SAP home page of your Integration Server installation.

Defining and Implementing a Local Correlation Function Module

Key fields from a business object will be used as the correlation ID by the Process Engine. If you deal with several business objects, their different key field values must be correlated to one common process instance ID generated by the Process Engine.

When implementing a local receiver function module, it is the responsibility of the ABAP programmer to set a meaningful value for field CORRELATIONOBJECTID from remote function module WM_PUSH_EVENT for all steps from the same process instance. There are two approaches to achieve this goal. You could either determine a meaningful correlation ID value directly as part of the local receiver function module implementation or assign a preprocessing correlation function module to the event-receiver coupling.

In the latter approach, the correlation function module populates the event container with additional document flow information. The local receiver function module then extracts this information from the event container to be used as input value for field CORRELATIONOBJECTID from remote function module WM_PUSH_EVENT. The sample Order to Cash process follows this approach.

Step 1: Define the Local Correlation Function Module

To correlate either predefined or custom events, you must define one or more local correlation function modules in your SAP system. You can define a local correlation function module to alter the event container of a specific event, many different events, or all the events you will use to push data to Optimize. You can define as many local correlation function modules as you want to alter the event container.

Any local correlation function module you define must implement the signature, as shown below:

```

*-----
*""Local Interface:
*  IMPORTING
*    VALUE (EVENT)           LIKE      SWETYPESCOU-EVENT
*    VALUE (RECTYPE)        LIKE      SWETYPESCOU-RECTYPE
*    VALUE (OBJTYPE)        LIKE      SWETYPESCOU-OBJTYPE
*    VALUE (OBJKEY)         LIKE      SWEINSTCOU-OBJKEY
*    VALUE (EXCEPTIONS_ALLOWED) LIKE    SWEFLAGS-EXC_OK DEFAULT SPACE
*  EXPORTING
*    VALUE (REC_ID)         LIKE      SWELOG-RECID
*  TABLES
*    EVENT_CONTAINER        STRUCTURE SWCONT
*  EXCEPTIONS
*    TEMP_ERROR
*    ANY_ERROR
*-----

```

The local receiver function module's signature should conform to function module SWE_TEMPLATE_CHECK_FB that is included in the SAP standard system installation. You can create a local receiver function module by creating a copy of SWE_TEMPLATE_CHECK_FB.

Step 2: Implement the Local Correlation Function Module to Alter the Event Container

During run time, event data are delivered to the correlation function module you defined in the previous step. The data are passed in a standard event container that includes the following elements:

- Event object type
- Event name
- Event object instance
- Event object key
- Event initiator (that is, the person under whose user ID the event was created)
- Event creation date/time

If an event has additional parameters defined with it, the event container will also contain these elements.

Beyond the additional event parameters available in the event container, you can add more fields to the event container. To set field CORRELATIONOBJECTID in the local receiver function module at a later time, retrieve a previously processed business object instance from the document flow and add it to the event container.

For the sample Order to Cash process, the process could be started by the Customer Inquiry Created, Customer Quotation Created, or Sales Order Created step:

- If the process-start step is Customer Inquiry Created, use the inquiry number as the correlation ID for the next step. You could also use the inquiry number as the correlation ID for all subsequent steps in one process instance.
- If the process-start step is Customer Quotation Created, use the quotation number as the correlation ID for the next step. You could also use the quotation number as the correlation ID for all subsequent steps in one process instance.
- If the process-start step is Sales Order Created, use the order number as the correlation ID for all subsequent steps in one process instance.

Generally speaking, when remembering all previously used document numbers as correlation ids for one process instance id at the Process Engine, it is sufficient to provide the document number from any previously processed document from the same document flow as the value for CORRELATIONOBJECTID with remote function module WM_PUSH_EVENT.

In your correlation function module, you would call a method that determines the previously processed document from the document flow and then updates the event container with the retrieved business object instance.

```
*****
FUNCTION z_wm_correlate_documents.
*"-----
  DATA hcl TYPE REF TO zcl_bem.
```

```

*- Initialize helper class
CREATE OBJECT hcl
  EXPORTING
    evt_container = event_container[]
    rectype       = rectype.

...

*- Add entry CORR_OBJECT to the container
CALL METHOD hcl->set_corr_object( ).
*- Retrieve the altered container
event_container[] = hcl->get_evt_container( ).
ENDFUNCTION.

```

In your local receiver function module you would then read the previously stored business object instance from the event container and set the input value for field CORRELATIONOBJECTID from remote function module WM_PUSH_EVENT accordingly.

```

METHOD push_event.
DATA : evtattr          TYPE sweqconts,
      timestamp         TYPE char15,
      bo_id             TYPE swotobjid,
      bo                TYPE obj_record,
      correlationid     TYPE char10,
      error             TYPE zbem_error.

...

*- Get correlation id
swc0_get_element _evt_container zbem_corr_object bo_id.
IF NOT bo_id IS INITIAL.
  swc0_object_from_persistent bo_id bo.
  swc0_get_object_key bo correlationid.
ENDIF.

*- As we report complete SAP LUW only, the start and stop timestamps
*- are usually very likely to be identical

*- Send event
CALL FUNCTION 'WM_PUSH_EVENT' DESTINATION zbem_dest
  EXPORTING
    eventname           = _evt_name
    eventobjecttype     = _evt_objtype
    receivertype        = _rectype
    eventobjectid       = _evt_objkey
    correlationobjectid = correlationid
    timestamp_start     = timestamp
    timestamp_stop      = timestamp
    businessdata        = _businessdata
    error_type          = error-type
    error_message       = error-message
    errormessagedetail = error-messagedetails.
ENDMETHOD.

```

Important: Before testing or running a local event-receiver coupling, you must link the receiver and correlation function modules with an event type by creating and enabling a matching adapter notification. For complete instructions, see ["Creating Remote or Local Event-Receiver Coupling Adapter Notifications" on page 41](#). You will specify the name of your local receiver function module in the **Receiver FM** property and your local correlation function module in

the **Correlation FM** property on the **Event-Receiver Coupling** tab on the adapter notification page.

5 Viewing Business Data

- Viewing Optimize for SAP Business Data 72

Viewing Optimize for SAP Business Data

After modeling an SAP business process and uploading it to the Process Engine for execution, the Analytic Engine analyzes the data received from SAP based on the KPI definitions and creates KPI instances.

At run time, you can view the KPI summary and instance information using My webMethods:

- Use the KPI Summary page to view and analyze the performance of KPIs and compare the performance of up to five KPIs on one graph. You can analyze historical performance. You can examine events such as rule violations as you analyze and compare KPIs. Based on this analysis, you can improve your business processes. For example, you can analyze order processing to identify and eliminate inefficient processes.
- Use the Process Instance Detail page to verify data for a single instance.

The instructions below provide a general overview of viewing the data based on the sample Order to Cash. For complete information about the Analytics Overview page, the KPI Summary page, and the KPI Process Instance Detail page, see *webMethods Optimize User's Guide*.

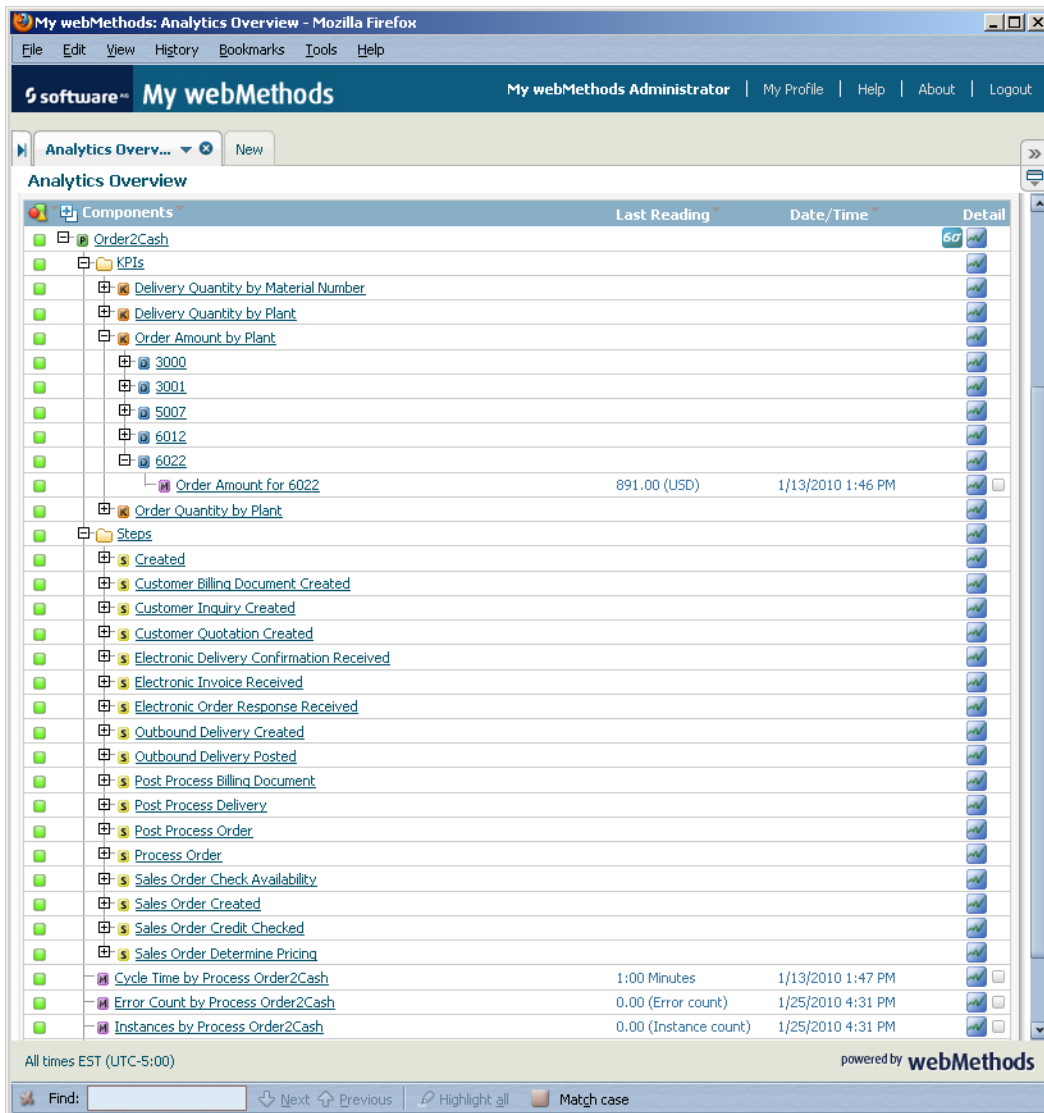
Viewing KPIs for Optimize for SAP


To view KPIs

1. In My webMethods: **Navigate > Applications > Monitoring > System-wide > Analytics Overview**.

The Analytics Overview page is displayed. This page enables you to select KPIs and view more information about them on the KPI Summary page. The Status column to the left of each component in the Analytics Overview panel indicates its current status. Additional icons beside each component indicate whether that component is a KPI, KPI instance, or dimension.

Figure 14. Sample Analytics Overview

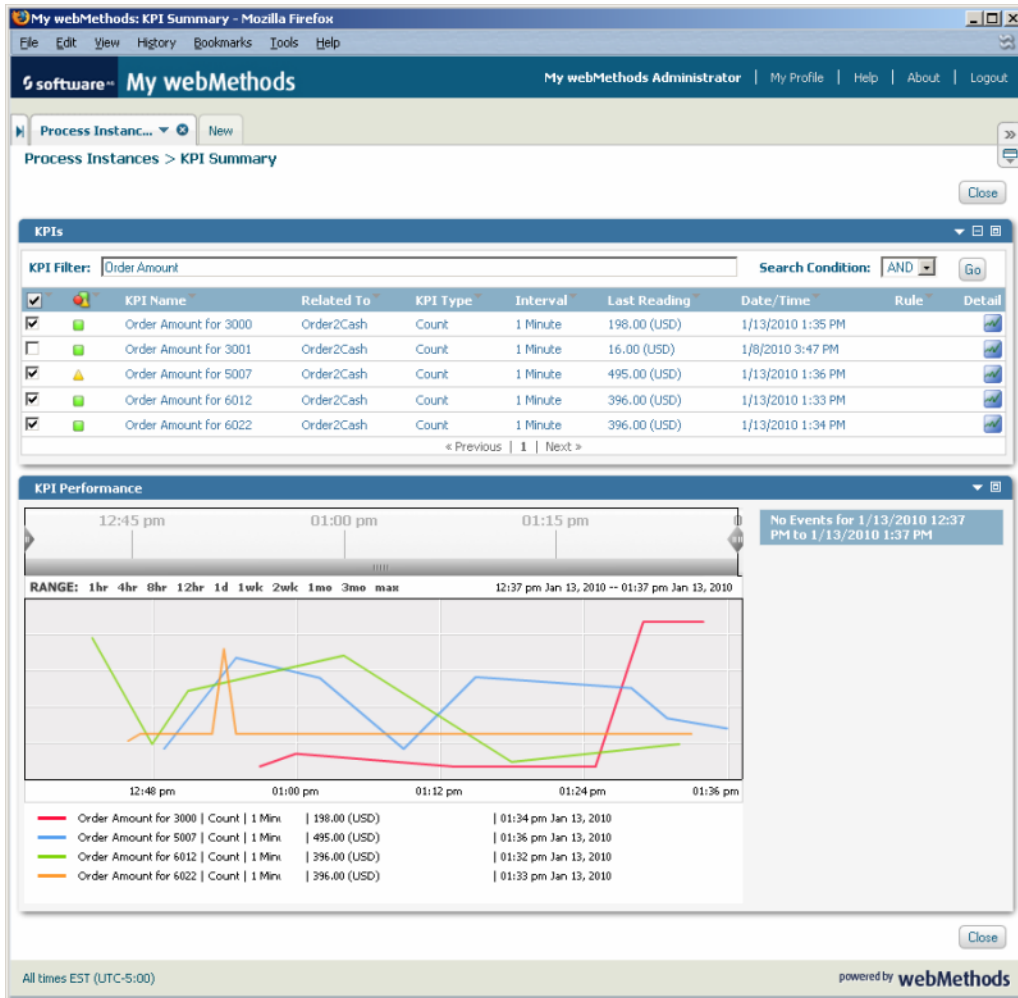


- You can drill down into each KPI and click  to view its KPI Summary page.

The KPI Summary page displays a list of all KPIs associated with the business process as shown in the sample below. Until you select KPIs, the KPI performance chart is blank.

- To fill in the KPI performance chart, set the KPI filter by entering the desired KPI name in the **KPI Filter** box. Select the check boxes in the column to the left of each desired KPI, and the KPI performance graph will get updated accordingly.

Figure 15. Sample KPI Summary



Viewing Process Instance Details

To view details for a single process instance

1. Click **Navigate > Applications > Monitoring > Business > Process Instances**.
2. Click the icon to the right of the corresponding Process Instance ID.

The Process Instance Detail page is displayed, as shown below:

Figure 16. Sample Process Instance Detail

Process Instance Information

Process Order/Cash
 Model Version 2
 Start Date / Time 1/7/2010 2:08:24.187 PM
 Last Updated 1/7/2010 2:12:54.507 PM
 Instance ID 75478208-6066-425e-9342-cde74da012b5
 Instance Iteration 1
 Status Completed
 Duration 0d 00:04:30.320

Step Summary

Start Date / Time	Last Updated	Instance Iteration	Step Name	Step Iteration	Status	Duration	Subprocess	Detail
1/7/2010 2:10:54.027 PM	1/7/2010 2:11:24.400 PM	1	Electronic Delivery Confirmation Received	1	Started	0d 00:00:30.373		
1/7/2010 2:10:23.667 PM	1/7/2010 2:10:54.027 PM	1	Outbound Delivery Created	1	Completed	0d 00:00:30.360		
1/7/2010 2:10:23.667 PM	1/7/2010 2:10:54.027 PM	1	Outbound Delivery Created	1	Started	0d 00:00:30.360		
1/7/2010 2:10:23.667 PM	1/7/2010 2:10:23.667 PM	1	Post Process Order	1	Completed	0d 00:00:00.000		
1/7/2010 2:10:23.667 PM	1/7/2010 2:10:23.667 PM	1	Post Process Order	1	Started	0d 00:00:00.000		
1/7/2010 2:09:54.340 PM	1/7/2010 2:10:23.653 PM	1	Electronic Order Response Received	1	Completed	0d 00:00:29.313		
1/7/2010 2:09:54.340 PM	1/7/2010 2:10:23.653 PM	1	Electronic Order Response Received	1	Started	0d 00:00:29.313		
1/7/2010 2:08:24.203 PM	1/7/2010 2:09:54.327 PM	1	Process Order	1	Completed	0d 00:01:30.124		
1/7/2010 2:08:24.203 PM	1/7/2010 2:09:54.327 PM	1	Process Order	1	Started	0d 00:01:30.124		
1/7/2010 2:09:54.293 PM	1/7/2010 2:09:54.293 PM	1	Sales Order Check Availability	1	Completed	0d 00:00:00.000		

11 - 20 of 29 Total

Control Actions

Activity Messages

Logged Fields

Date / Time	Step Name	Instance Iteration	Step Iteration	Input/Output	Field Name	Field Value
1/7/2010 2:10:54.027 PM	Outbound Delivery Created	1	1	Output	eventObject/fields/createdSets/WERKS	0012
1/7/2010 2:10:54.027 PM	Outbound Delivery Created	1	1	Output	eventObject/fields/createdSets/MATNR	DEMO-SP-0012
1/7/2010 2:10:54.027 PM	Outbound Delivery Created	1	1	Output	eventObject/fields/createdSets/LFIMG	11
1/7/2010 2:10:23.653 PM	Electronic Order Response Received	1	1	Output	ORDERS05/IDOC/E1EDP01/WERKS	0012
1/7/2010 2:10:23.653 PM	Electronic Order Response Received	1	1	Output	ORDERS05/IDOC/E1EDP01/NETWR	363
1/7/2010 2:10:23.653 PM	Electronic Order Response Received	1	1	Output	ORDERS05/IDOC/E1EDP01/MENGE	11

1 - 6 of 6 Total

Process Errors

Date / Time	Error	Error Message	Message Detail	Step Name	Step Iter	Service Name	Server ID
No Errors for this Process Instance							

All times EST (UTC-5:00) powered by webMethods

- To view the KPI Summary page for the business process, click the icon.

Analyzing System Resource Monitoring Data

Use the My webMethods user interface to view and analyze monitoring data, as follows:

- Find resources that out of compliance and view information about the rules that were violated.
- View a high-level summary of all resources and the performance of the entire enterprise.

- Compare the performance of different KPIs for resources and analyze historical KPI performance to find positive or negative trends.
- View alerts about resources that Optimize predicts will go out of compliance in the future.
- View reports about KPIs for system resources.