SAP NetWeaver™ in the Real World

“SAP NetWeaver in the Real World” is a five-part series of technical articles demonstrating a step-by-step implementation of SAP NetWeaver and its key components. The business scenario of the fictional Iridium Motors showcases the components of the SAP NetWeaver stack, and the real-world utility achieved by their integration.
# Table of Contents

SAP NetWeaver™ in the Real World ..................................................... 1  
Table of Contents ............................................................................ 2  
Introduction ..................................................................................... 3  
Scenario ............................................................................................ 4  
About the SAP Exchange Infrastructure ........................................... 5  
Integration ....................................................................................... 5  
Prerequisites .................................................................................... 8  
Terminology/Concepts ..................................................................... 9  
Implementation ............................................................................... 10  
Overview of the Scenario ................................................................ 10  
System Landscape Directory ............................................................ 11  
  Creating a New Product and Software Component Version for the Sender  
  and Receiver Systems .................................................................... 11  
  Step-by-Step Solution .................................................................... 13  
  Define the Technical and Business Systems in the System Landscape  
  Directory ....................................................................................... 15  
Modeling Using the Integration Repository ......................................... 27  
  Design Objects in the Integration Builder ........................................ 28  
  External Programs and Descriptions .............................................. 29  
  Step-by-Step Solution .................................................................... 30  
Configuring the Scenario in the Integration Directory ......................... 48  
  Integration Directory Principles ..................................................... 48  
  Enter Iridium Motors .................................................................... 53  
  Step-by-Step Solution .................................................................... 54  
Executing the Scenario ..................................................................... 65  
Preview of “SAP NetWeaver™ in the Real World” Part IV .................... 69
Introduction

In "SAP NetWeaver in the Real World" Part III, Iridium Motors developers use SAP® Exchange Infrastructure (SAP XI) to configure the data communications between the manufacturing system and the SAP R/3 based On-line Transaction Processing (OLTP) system, enabling John, the production operator, to transmit, with a simple push of a button, the equipment failure data to the manufacturing system and trigger the proper shut-down of equipment.
Scenario

09 September 2004
Iridium Motors, Inc.- Dallas Texas

11:25 a.m.
An alarm sounds on the shop floor of Iridium Motors. A high-speed motor suddenly breaks down. John, the production operator, immediately punches the red, flashing “Equipment Failure” button. As a result, all critical equipment in the production unit shuts down as a precautionary measure. The entire production process grinds to a halt.

John rushes to inform the production manager, Anne, about the equipment failure. Anne pages the maintenance supervisor, Raul.

12:15 p.m.
After meeting with Anne to discuss the situation, Raul collects all the information he can about the equipment failure from John. Back in his office, he begins to sort through relevant documentation on his desktop computer. He pulls up various files, wading through manuals and maintenance records to find patterns that would serve as a reference point for carrying out some basic troubleshooting. He also calls the equipment manufacturer to request that service technicians come as soon as possible. Finally, he passes along the troubleshooting information to John.

3:35 p.m.
John returns to the shop floor with an assortment of troubleshooting materials, which he then begins applying to the malfunctioning motor.

---

1 The names and events in SAP NetWeaver™ in the Real World are fictional. Any similarity to real people, companies, or events is merely coincidental.
About the SAP Exchange Infrastructure

The SAP® Exchange Infrastructure (SAP XI) is the SAP platform for process integration based on the exchange of XML messages. It enables you to connect systems from different vendors (both SAP and non-SAP), different versions, and different programming languages (Java, ABAP, etc.).

SAP Exchange Infrastructure is based on an open architecture, uses open standards (specifically those from the XML and Java environments), and offers services essential in a heterogeneous and complex system landscape:

- Runtime infrastructure for message exchange
- Configuration options for managing collaborative processes and message flow
- Design and execution of cross-component business processes
- B2B support
- Options for transforming message contents between sender and receiver systems

Integration

The following graphic represents the positioning of SAP XI within the SAP NetWeaver™ stack, with the key XI components illustrated on the right.
To facilitate the integration of external systems, SAP XI is based on general standards. At the center of the infrastructure is an XML-based communication system that uses HTTP (Hyper Text Transfer Protocol). The application-specific contents are transferred from sender to receiver in messages [based on a user-defined XML (eXtensible Markup Language) schema] using the Integration Server.

Senders and receivers exchanging messages via the Integration Server are in different places; however, this actually makes it easier to connect systems using different technologies. Every system with the ability to exchange messages with the Integration Server also has the ability to exchange messages with all other systems connected to the Integration Server. SAP XI supports the following methods of communication with the Integration Server:

- Direct communication using proxies, which are generated in the application systems using a description in WSDL (Web Service Description Language)
- Communication using adapters – in this case, interfaces are created for message exchange in the application system, or use existing interfaces

Simple message processing on the Integration Server is stateless; i.e., the Integration Server does not make any semantic connections between various messages. Cross-component business processes, which are executed on the Integration Server, are used to describe logical dependencies between messages and define additional conditions for the message flow. SAP XI is used to model and centrally maintain these cross-component Business Process Management processes.

As with cross-component business processes, the entire integration knowledge of a collaborative process is saved centrally in SAP XI. Objects created at design time are saved in the Integration Repository and objects created at configuration time are saved in the Integration Directory. In this way, SAP XI follows the principle of shared collaboration knowledge; that is, it is better to search for information about a collaborative process centrally rather than in each of the systems involved. This procedure considerably reduces the cost of developing and maintaining the shared applications.

The following graphic represents the building blocks of SAP XI.
Prerequisites

Installed Software

To implement the solution described in this article, you need SAP® Web Application Server 6.40 for Java and SAP Exchange Infrastructure 3.0 installed in the Web Application Server (WebAS). In order to access the XI Integration Builder, you need to have Java™ WebStart version 1.4.2 installed on the local computer.

https://www.sdn.sap.com/sdn/developerareas/was.sdn?page=software_download.htm

Download Java 2 Platform, Standard Edition, v 1.4.2 (J2SE)
http://java.sun.com/j2se/1.4.2/download.html

It is further assumed that the reader has a working installation of SAPGUI and a valid user id to log on to an SAP R/3 system with sufficient authorizations to perform an RFC call.
Terminology/Concepts

System Landscape Directory
Certain objects—namespaces, software components, and products—are contained in the system landscape directory. From this point of view, the System Landscape Directory is an integral part of the content development. It also contains the details of the systems within a landscape.

Integration Repository
Objects defined in the integration repository are defined during the development phase of the project and are part of the business package that can be delivered to the customer.

Integration Directory
Objects set up in the integration directory must be defined during the implementation phase for each customer. They are not part of the delivery through the business package.

Integration Server
This is the runtime environment for the SAP XI. The message transaction happens within the Integration Server. It controls the engines that are required for carrying out the message. It uses the integration knowledge built using the Integration Repository and Integration Directory.
Implementation

Overview of the Scenario

The manufacturing system triggers an equipment failure. The failure message is sent through SAP Exchange Infrastructure (SAP XI) to the SAP R/3 based OLTP system, changing the equipment status to a failure state. This is achieved by triggering the execution of Remote Function Call (RFC) present in the OLTP system from the SAP XI integration server.
System Landscape Directory

The SAP System Landscape Directory (SLD) is the central information provider in a system landscape.

The SLD contains two types of information:

- **Component information**: This includes information about all available SAP products and components, including their versions. If there are any third-party products in the system landscape, they are also registered here.
  At design time of the integration objects, the component information is extracted from the SLD to define business scenarios. (The integration objects—namely, data types, message interfaces, message mappings, interface mappings—are configured using the Integration Repository.)

- **Landscape description**: This contains all installed systems in a system landscape. When a collaborative business process is configured, the landscape description is needed to determine the system information of the business partners involved.

Creating a New Product and Software Component Version for the Sender and Receiver Systems

The first step in the SLD is to configure the components (products and software component versions) for the interacting applications in the System Landscape Directory.

- **Product**: The version of a product is the unit that can technically be delivered to the customer. Each product version contains one or more software component versions.
- **Software Component**: The version of a software component is the smallest unit that can be delivered as a product. Each software component contains multiple namespaces and its corresponding integration objects.

The following graphic illustrates how the products and software component versions are related to each other.
Relationship of Products and Software Component Versions

We will create two products and two software component versions for our scenario, one for sending and one for receiving applications. The details are as given below:

**Product and software component version for the sending application:**

Product
Vendor: SAP Demo
Name: Plant_Maintenance_Sender
Version: 1.0

Software Component Version
Vendor: SAP Demo
Name: PLANT_MAINTENANCE_SENDER
Version: 1.0

**Product and software component version for the receiving application:**

Product
Vendor: SAP Demo
Name: Plant_Maintenance_Receiver
Version: 1.0

Software Component Version
Vendor: SAP Demo
Name: PLANT_MAINTENANCE_RECEIVER
Version: 1.0
Step-by-Step Solution

1.1. Open the Integration Builder (Exchange Infrastructure -> Start Integration Builder)

1.2. Click on the System Landscape Directory link.

1.3. You will see the products and its related software components in the Software Catalog. Click on New Product....
1.4. Enter the name of the vendor, product, and the version corresponding to the sender system. Click *Create*.

![Define Product](image1)

**Define Product**

Add a non-SAP product to the software catalog.

- **Vendor:** SAP Demo
- **Name:** Plant_Maintenance_Sender
- **Version:** 1.0

[Create] [Cancel]

Namespace: sld/active  X3A  Object Server: PWDF0321

1.5. Enter the details for the software component version. Click *Create*.

![Define Software Component](image2)

**Define Software Component**

Add a software component to a non-SAP product.

- **Product:** Plant_Maintenance_Sender, 1.0 of SAP Demo
- **Vendor:** SAP Demo
- **Name:** PLANT_MAINTENANCE_SENDER
- **Version:** 1.0

[Create] [Cancel]

Namespace: sld/active  X3A  Object Server: PWDF0321

1.6. Repeat steps 1.3 to 1.5 for the Receiver system.
Define the Technical and Business Systems in the System Landscape Directory

The second step in the SLD is to register the systems that are present in the landscape.

Technical systems
Technical systems are the technical representation of the systems that are present in the landscape. They can be either SAP or third-party systems. These technical systems will typically have the products and software component versions that are created in the Software Catalog installed in them.

Business Systems
Business systems are logical systems that communicate with each other within SAP Exchange Infrastructure by sending and receiving messages. They can be either SAP or third-party systems.

- An SAP system has one or more clients that function independently of each other as logical units at runtime. Each of these clients represents a business system in SAP Exchange Infrastructure.
- A third-party system is also a logical unit that functions as a sender or receiver. Therefore, third-party systems are also business systems in this sense.

Business systems are logical representations of the technical systems. They also identify the related SAP Exchange Infrastructure integration server.

In our scenario, there are two technical systems and two business systems that are representative of the sender and receiver systems.

The following are the Technical system details:
- A third party technical system, “Plant_Maintenance_Sender,” corresponding to the sender system
- A WebAS ABAP technical system, “Plant_Maintenance_Receiver,” corresponding to the receiver system

The following are the Business system details:
- A third party business system, “Plant_Maintenance_Sender,” corresponding to the sender system
- A WebAS ABAP business system, “Plant_Maintenance_Receiver,” corresponding to the receiver system
2. Define the Technical Systems in the System landscape Directory

2.1 From the System Landscape Directory, click the Technical Landscape link.

2.2. Click New Technical System…
2.3. In the Technical System Wizard, select the type of technical system you want to create. In our scenario, the sending system is treated as a third-party system.

2.4. Provide a System Name and a Host Name. The host name may be given as "localhost" or the name of the machine that will act as the third-party system.
2.5. Add software products and components, i.e., the product created for the sender system.

2.6. To enter the system details for the receiving system, click *New Technical System*....
2.7. The receiving system in our scenario is an R/3 based OLTP system (Web AS ABAP) system.

2.8. Enter the system name, installation number, and database host name. The details of the Installation Number can be retrieved from the WebAS ABAP system using the transaction “slicense.” The Database Host Name of the WebAS ABAP system can be retrieved from the SAP logon pad.
2.9. Enter the Host Name and Message Port under Message Server and the information for Application Server.

![Technical System Wizard - Message Server and Central Application Server]

2.10. Optional: The following information needs to be added if you need to define additional application servers.

![Technical System Wizard - Additional Application Server (optional)]
2.11. Enter the Client Number and click *Add*.

2.12. Choose the product we created for the receiver system.
3. Define the Business Systems in the System Landscape Directory

3.1. From the System Landscape Directory, click the Business Landscape link.

3.2. Click *New Business System*....

3.3. Enter an appropriate business system name.
3.4. Create a third-party business system to represent the sender.

3.5. Choose the technical system that we created for the sender. The Logical System Name field can be left blank.

3.6. This screen will prompt you to choose from the available products that are linked to the technical system. Choose the product we created for the sender.
3.7. Choose the related SAP XI integration server which will handle the message flow.

![Business System Wizard - Integration Server](image)

3.8. Click *New Business System*....

![Business Landscape](image)

3.9. Enter an appropriate business system name for the receiver.

![Business System Wizard - Details](image)
3.10. The business system for the receiver will be a Web AS ABAP system.

3.11. Choose the technical system that we created for the receiver.

3.12. This screen will prompt you to choose from the available products that are linked to the technical system. Choose the product we created for the receiver system.
3.13. Choose the related SAP XI Integration server which will handle the message flow.
Modeling Using the Integration Repository

The Integration Repository is part of the Integration Builder tool that helps in building the design-time knowledge.

Communication in SAP Exchange Infrastructure is interface-based. That is, messages are generally created by calling an interface. SAP Exchange Infrastructure supports two approaches for implementing a cross-system process:

- **Outside-In**: The relevant interfaces for the cross-system process are created in the Integration Builder. These *message interfaces* are a programming language-independent description in XML. These message interfaces are used to generate callable interfaces (*proxies*) in target systems.

- **Inside-Out**: The functions that are to be called using SAP Exchange Infrastructure already exist in the application systems. To use an interface description of these functions in the design process, import specific interface descriptions to the Integration Repository by using the Integration Builder (see *External Programs and Descriptions* at the end of this section).

The two approaches can also be combined. Regardless of the interfaces that are required for your cross-system process, you must work simultaneously in the following development environments during design time:

- Use the Integration Builder to design and define all objects affecting message exchange (see below). In some cases, you can use external tools and import objects to the Integration Repository.

- Implement message processing in your application program by using the development environment of your application systems.
Design Objects in the Integration Builder

The following graphic provides an overview of the design objects that developers can create in the Integration Repository by using the Integration Builder.

Objects Created by the Integration Builder

The Integration Builder (Integration Repository) allows you to edit the following objects by using the corresponding graphical editors (shown on the left-hand side of the graphic):

- **Business scenarios** describe the communication between application components on a higher level of abstraction. **Business processes** are executable processes on the Integration Server.

- **Mappings** are used to define structure or value mappings between messages that are exchanged between interfaces. These mappings can be defined as graphical message mappings or they can be imported into the Integration Repository as XSLT or Java archives.

- **Context objects** mask access to elements or attributes in the payload. For example, elements can be specified in a deeply nested message structure by using the short name of a context object, thus sparing you the long hierarchy path.

- **Data types** and **message types** describe the structure of messages that are to be exchanged using **message interfaces**. Developers use message interfaces to generate proxies in application systems.

The entire content of the Integration Repository can be shipped. Together, these objects are referred to as **Process Integration Content**, abbreviated as **XI content**. Using a **software component version** from the System Landscape Directory, the smallest possible
shipment unit is defined for a number of objects that belong together in the Integration Repository. SAP software component versions are also the basis for shipment units from application objects in SAP systems, so that XI content and application content can be assigned to a joint software component version in the SAP system.

**External Programs and Descriptions**

SAP XI uses various XML standards. The advantage of this is that you can reuse external programs and schema definitions. The following graphic gives an overview of the formats that you can import to and export from the Integration Repository.

![Integration Repository Diagram](image)

**Importing and Exporting Formats from the Integration Repository**

- You can import externally defined business processes as BPEL documents (Business Process Execution Language).
- As an alternative to graphical message mappings (which the Integration Builder uses to generate Java programs), you can also import externally defined XSLT mappings or Java mappings to the Integration Repository to execute at runtime. It is also possible to combine different mapping program types with each other.
- You can import interface descriptions for IDocs and RFCs.
- You can import external definitions (DTD, XSD, and WSDL) and reuse message schemas from these documents within the Integration Builder.
Step-by-Step Solution

The design time configurations are done for the software component versions. To view and model the data types, message mappings, etc., we first need to import the software component versions from the System Landscape Directory that correspond to the sender and the receiver systems.

- Get to the SAP Exchange Infrastructure main page and navigate to the Integration Repository link.
- The main page of the “Design: Integration Builder” has the menu option Tools. Navigate to Tools->Transfer from System Landscape Directory->Import Software Component Versions….
- Choose the Software Component Versions that correspond to the sender and the receiver systems and choose Import.
- After a successful import, the software component versions are displayed in the tree on the left hand side of the screen, with which we can model the design time configurations like data types, message mappings, etc.
For our scenario, we need to create the following in the software component (PLANT_MAINTENANCE_SENDER) that corresponds to the sender system:

1) Since the sender system is a third party system, the “Interface Import” should be checked to “Not Permitted.” We do not have any RFC or IDOC imports possible in this case.

2) Enter a namespace (http://sap.com/demo/plantmaintenance) in the Namespaces section. Repository namespaces are used to avoid naming conflicts within object types in the Integration Repository.
Once a namespace is created, you can create the repository objects such as data types, message types, message mappings, etc. for a particular namespace in a software component version.

Note: A software component version can have number of namespaces.

- Create a data type (PlantMaintenanceDatatype) containing the following information:
  1. Equipment ID (Type: String)
  2. Functional Location (Type: String)
  3. Date (Type: Date)
  4. Time (Type: Time)

- Create a message type (PlantMaintenanceMessagetype) for the data type that was created in previous step.

- Create a message interface (PlantMaintenanceMessageInterface) of type “Outbound” with the message type that was created in the previous step. Interfaces types are specific to the systems. Since the sender system will send (outbound from the system) a message, the interface type is set to “Outbound”.
For our scenario, we would need to create the following in the software component (PLANT_MAINTENANCE_RECEIVER) that corresponds to the receiver system:

1) Since the receiver system is a SAP WebAS ABAP system and we plan to import an RFC interface, the “Interface Import” should be checked to “Import of RFC and IDOC Interfaces from SAP Systems Permitted”.

2) In the “Connection Data for Import from SAP System”, give the system details of the SAP WebAS ABAP system that acts as the receiver system.

3) Enter a namespace (http://sap.com/demo/plantmaintenance) in the namespaces section. Repository namespaces are used to avoid naming conflicts within object types in the Integration Repository. SAP XI assumes in most cases that it can identify an object uniquely (at least within one system) from its name and repository namespace.
Once a namespace is created, you can create the repository objects like data types, message types, message mappings etc for a particular namespace in a software component version.

Since the receiving system is a SAP WebAS ABAP system and we intend to execute an RFC in the receiving system, the message interface will be that of an RFC. The RFC can be imported be as follows:
Provide the user credentials to login to the system, details of which are provided in the “Connection Data for Import from SAP System” section. Once connected the screen will list the IDOC’s and RFC’s that can be imported from the system. For our scenario we need to import the RFC interface BAPI_EQMT_DISMANTLEFL.
The next step is to create a mapping between the PlantMaintenanceMessagetype and BAPI_EQMT_DISMANTLEFL.

Create a new Message Mapping called PlantMaintenanceMessageMapping, with the source as the PlantMaintenanceMessagetype and the destination as BAPI_EQMT_DISMANTLEFL. Drag and Drop the source field to the destination field to do a field-to-field mapping. Also, map the header PlantMaintenanceMessagetype to BAPI_EQMT_DISMANTLEFL.
Note: The message mapping could be tested with some dummy values by executing it in the “Test” tab.
Once the message mapping is done, we need to create an interface mapping between PlantMaintenanceMessageInterface and BAPI_EQMT_DISMANTLEFL, and specify the mapping program as the message mapping just created.

After we have done the configurations, we need to activate the changelist. We will have two changelists corresponding to the two software components that we have edited: PLANT_MAINTENANCE_SENDER and PLANT_MAINTENANCE_RECEIVER. We need to activate both these changelists. The following diagram shows the activation of one of the changelists.
Configuring the Scenario in the Integration Directory

Integration Directory Principles

The Integration Directory captures the configuration time knowledge required for two parties/systems/services to interact with each other. The information is captured as configuration objects in the Integration Directory.

The configuration describes how the Integration Server is to process inbound messages and to which receiver or receivers messages must be sent. The information captured in the Integration Directory falls into three cross-component areas, shown in the table below.

<table>
<thead>
<tr>
<th>Area in the Integration Directory</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaboration Profile (Objects: communication party, service, communication channel)</td>
<td>In the collaboration profile, you document the technical options available to the communication parties for exchanging messages. Specify the potential senders and receivers of messages and the technical communication paths.</td>
</tr>
<tr>
<td>Logical Routing (Objects: Receiver determination, Interface determination)</td>
<td>In logical routing, you define the flow of messages in a system landscape.</td>
</tr>
<tr>
<td>Collaboration Agreement (Objects: Sender agreement, receiver agreement)</td>
<td>In collaboration agreements, you define the technical details for message processing (for example, adapter configuration) and the security settings for specific sender/receiver pairs.</td>
</tr>
</tbody>
</table>

All the information stored in the Integration Directory is available in the Integration Server (and the Adapter Engines, if they are involved in the message processing) as runtime cache. At runtime, the information is interpreted by the Integration Server and executed.

We will look into these areas in detail.
As mentioned, the *Collaboration Profile* is the collective name for the group of objects in which the technical details required for the two parties or systems to communicate are captured.

In case of a partner-to-partner communication (as in B2B scenarios), we created *parties* with necessary attributes to identify (alternate identifiers) the communicating parties.

The key object that needs to be set up for a collaboration profile is the *service*, which can be one of the following:

- Business Service – Created in the directory to represent an application interface
- Business Process – A business process imported from the repository
- Business System – A business system imported from the SLD
A service represents the application system interface that receives messages or sends out messages in the XI world. Each service is assigned one or more message interfaces (directly or indirectly) that define the syntax of the message that the service receives (Outbound Interface) or sends out (Inbound Interface). Communication channels are also created for services, defining the communication rules, the technical channel (protocol, standard etc.) and the technical end-points.

A service may or may not be associated to a party.

Since the configuration of channels can be quite extensive, it is possible to use channel templates to support the configuration of channels.

**Logical Routing**

Once a message reaches the Integration Server, the message has to be routed to the target system (represented as a service in the Integration Directory). This is achieved using the Logical Routing.

Figure 3.1.b shows an overview of the objects involved in the logical routing, which is handled in two steps:

- Receiver Determination
- Interface Determination

The incoming message will have the party, service, and the message interface information in it for the sender and the receiver (optionally). We need to determine the receiver party and the service for this message. This is achieved using a Receiver Determination. We can define one or more receivers for a message, and also use some conditions to identify the correct receiver.

Once the receiver service is identified, we might need to change the interface of the message to the interface that is expected by the receiver service. We define Interface
**Determination** for this, where we specify the target interface, and the **Interface Mapping** (if any) required to map the source interface to the target interface.

Logical routing is an optional step.

**Collaboration Agreements**
The final step required in the process flow is the collaboration agreements for the sender and for the receiver. Figure 4.1.c gives an overview of the objects involved in this step.

- In the *collaboration agreements* we determine the actual channels that will be used for communication purposes.
- In the *sender agreement* we determine the channel that will be used for the communication with the sender (the party sending the message to XI).
- In the *receiver agreement* we determine the channel that will be used for the communication with the receiver (the party receiving the message from XI).

![Collaboration Agreements](image)

Picture 3.1.c: Overview of the objects involved in the collaboration agreements

**Tip**: Please note that the nomenclature of the Collaboration Agreements and the message interfaces are always given with respect to the application. E.g., if a message is sent out by an application, the corresponding message interface will be an outbound message interface. We will create a Sender Agreement for this message to be routed to XI.
Figure 3.1.d shows the relationship between the various objects explained here.
Enter Iridium Motors

In the plant maintenance scenario at Iridium Motors, we have a MES system (PlantMaintenanceSender System) that sends an “Equipment Damaged Message” (PlantMaintenance Message Interface), which has to reach a BAPI that will update the status of the equipment as “damaged.” The MES system dumps the message as an XML message to a file system. We configure a file adapter to receive this file and route it to the BAPI using an RFC Adapter.

We will see in detail how to achieve this in the following section, in which we look into the step-by-step configuration of the scenario in the Integration Directory.
Step-by-Step Solution

Let us examine the configuration steps in the Integration Directory in detail.

We will first need to create an “Integration Scenario” in the Integration directory. The scenario is used to group the objects together. Then we will perform the following steps:

- Import business systems/create business services
- Create communication channels for the systems/services
- Define logical routing rules, i.e. receiver determination and interface determination
- Define the collaboration agreements – sender and receiver agreements, and assign the communication channels appropriately.

We can use the configuration wizard for doing this.

First launch the Integration Directory tool.
Create a new scenario in the Integration Directory.
We also have a feature to import a Business Scenario from the repository if we have configured it. In our case we have not configured a business scenario.

Save and activate the scenario.

We can now go to the Menu -> Tools - > Configuration Wizard and use this to configure our scenario.
Welcome to the Configuration Wizard

You use this program to define message processing on the Integration Server from the inbound channel to the outbound channel.

To define the processing of the message, you must describe the inbound and outbound messages.

You can select one of the following two configuration scenarios:

**Internal Communication**
Select this scenario if you want to exchange messages within your system landscape.

**Party Communication**
Select this scenario if you want to exchange messages with external partners.
We can use XI to exchange messages within our system landscape between applications (A2A Integration) or with other partners outside our systems landscape (B2B Integration).

We must choose the first option for configuring our scenario. In the next screen, choose the appropriate values for the Inbound Message. We can use the “value help” feature to select the appropriate values from the SLD (for business systems) and Integration Repository (for message interfaces).

When choosing the adapter type, we have to choose the file adapter only if we are using the file adapter on the J2EE Adapter Engine. In this case, we need to configure the communication channel and do the necessary configuration for the file adapter to pick the file from a source directory and send it to the XI system.

In our case, we are using a stand-alone J2SE File Adapter. Hence the configurations are done on the stand-alone file adapter. File adapter will add the XI-specific header elements to the file (payload) and send it the Integration server (the URL will be mentioned in the file adapter configuration). Hence, the inbound message is in the XI format and we choose the adapter type as XI.

Continue to the next step, where we must specify the details for the Outbound message.
Here, we choose the business system from the SLD using the value help, the BAPI interface that we imported in the repository and the adapter type as RFC. RFC adapter is used to call the BAPI from the XI system.

The next step is to create the Sender agreement. Since we have chosen the adapter type as XI, we need not create the communication channel and the Sender agreement for that. So, we can proceed to the next step where we specify the Logical Routing rules.
We create the Receiver Agreement and specify the channel to end the configuration. For business systems, Integration Builder created default receiver communication channels for RFC and IDoc adapters. We can use the GeneratedReceiverChannel_RFC in our case. We can generate all the objects and activate them.
Generate Objects

You have entered all the necessary information for your configuration. The configuration wizard can now generate the required objects.

To start generation of the objects, choose Finish.

Add to Scenario: PlantMaintenanceScenario
We can verify from the log whether all the objects were created properly. We need to provide the logon details in the RFC Communication channel that was created. After this, we can activate the changelist.
Executing the Scenario

Now that we have configured the scenario, we need to test it. We can use a J2SE File Adapter to test the scenario. We have to configure the File Adapter so as to send a sample XML message to the Integration Server. The reference configuration is given below. (For details of the Adapter Engine and File Adapter, please refer to the XI Documentation.)

```java
## file adapter java class
classname=com.sap.aii.messaging.adapter.ModuleFile2XMB
version=30
mode=FILE2XMB

## Integration Engine address and document settings (example, see docu)
XI.TargetURL=http://<host>:<port>/sap/xi/engine?type=entry
XI.User=XIAPPLUSER
XI.Password=XIPASS
XI.QualityOfService=EO
#XI.SenderParty=
XI.SenderService=Plant_Maintenance_Sender
#XI.ReceiverParty=
XI.ReceiverService=Plant_Maintenance_Receiver
XI.Interface=PlantMaintenanceMessageInterface
XI.InterfaceNamespace=http://sap.com/demo/plantmaintenance

##File Adapter specific parameters (example, see docu)
file.type=TXT
file.sourceDir=<source_directory>
file.sourceFilename=*
file.processingMode=delete
file.pollInterval=20

```

After saving the configuration, restart the File Adapter. Now, to send the message, we need to copy the XML payload into the source directory configured in the File Adapter. The sample XML message is shown below.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ns:PlantMaintenanceMessagetype
xmlns:ns="http://sap.com/demo/plantmaintenance">
  <EquipmentID>SLIEQUIP5</EquipmentID>
  <FunctionalLocation>KB</FunctionalLocation>
  <Date />
  <Time />
</ns:PlantMaintenanceMessagetype>
```
The file adapter will poll the directory. When it finds a file to be sent there, it will send it as an XI Message to the Integration Server URL mentioned in the configuration. We can see the File Adapter log that will show the processing steps like this:

14:46:51 (4051): Process 1 file(s):
14:46:51: c:\pmtest\FileInputPM.txt
14:46:51 (4052): Start processing "TXT" file "c:\pmtest\FileInputPM.txt" size 268 in "EO mode
14:46:51 (4062): Sending text message "d03f9370-062e-11d9-ac39-000bcd45f03d" type "application/xml" to Integration Engine...
(message size: 268 bytes, URL: "http://pwdf0321:50031/sap/xi/engine?type=entry")
14:46:54 (4037): File "FileInputPM.txt" processed successfully

Now, to see the message processing on the XI System in the Integration Engine, we can use the transaction SXMB_MONI on the XI System. The monitor shows that the message has been successfully processed. It will also show that the outbound processor for the message is an Adapter Engine (AENGINE as shown by the Monitor). The message would have gone through the various steps that we have configured in the Integration Builder, and will be delivered to the RFC Adapter that will call the BAPI, BAPI_EQMT_DISMANTLEFL. We can verify these details by double-clicking on the row in the monitor, which will open a details screen as shown below.
The BAPI will be executed and update the status of the Equipment SLIEQUIP5 (as mentioned in the test XML message that we sent) to “Equipment Damaged.”
Now, if we go to the target system configured in the communication channel, which is an SAP R/3 system, and execute the transaction IE03, we can see the status of the equipment will be updated from “0001” (Equipment Released to Production) to “0007” (Equipment Damaged.) This is demonstrated in the following screen shots from the systems.

![Change Status](image1)

![Display Equipment: General data](image2)
Preview of “SAP NetWeaver™ in the Real World” Part IV

The next article in the “SAP NetWeaver™ in the Real World” series is centered on SAP® Business Intelligence (BI). The development team at Iridium Motors will build a BI InfoCube to streamline equipment reporting processes.

The InfoCube, which contains historic data about the failed equipment, will provide information about its service history. When the Production Manager Anne notifies Raul, the maintenance supervisor, of the equipment failure, he can immediately launch BI reports to analyze past equipment problems and solutions. With information readily available about prior failures, average service time, and corrective actions taken, diagnosing and troubleshooting can be carried out without delay.

Raul and Iridium Motors are nearly up and running again!