Supporting SAP XI in SAP xMII 11.5

Applies to:

SAP xMII 1.5 SR2

Available download: SAPXI.zip

Summary

One of the key aspects of SAP xMII is the ability to serve as a connecting link between plant floor systems and the 'Corporate ERP world' of a mySAP ERP or an SAP R/3 system. With its flexible approach of modeling a business process as a 'Business Logic Transaction', combined with its visual capabilities and shop floor connectors, it is ideally suited to deal with the large amount of disparate shop floor systems. Since each plant tends to have its own, very specific, non-standardized business processes, using SAP xMII at the plant site is essential to reduce the complexity in integrating a plant site with an ERP system. However, SAP xMII's capabilities are more limited when it comes to supporting a larger system landscape, including several mySAP systems and a multitude of plant sites. Integrating such a large scale environment is a classical domain of the SAP NetWeaver Exchange Infrastructure, commonly know as SAP XI. Therefore, using a combined solution, in which SAP xMII handles the integration with individual shop floor systems, and SAP XI acting as central message broker for the overall landscape, provides you with the best of both worlds.

The goal of this article is to provide you with the code samples and content to use SAP xMII as this 'extension' of SAP XI into the plant sites.

Author(s): Stephan Boecker

Company: SAP

Created on: 18 March 2007

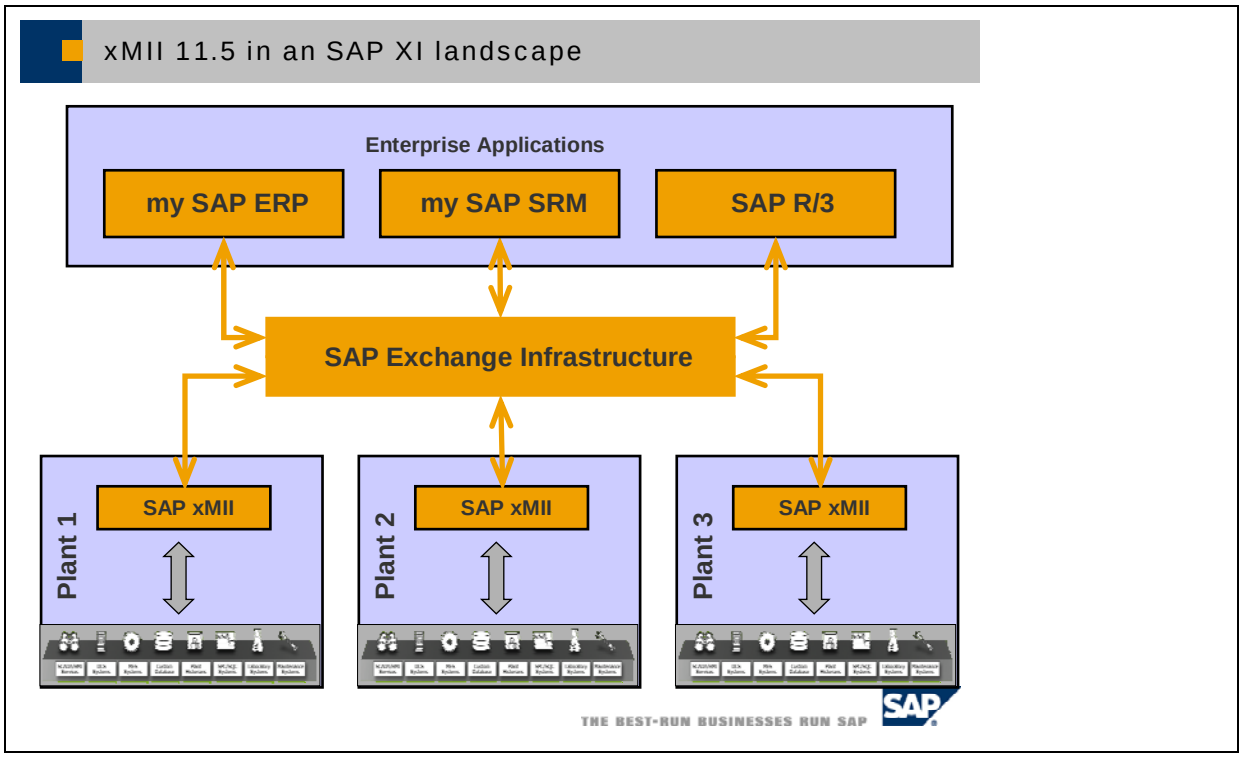
Author Bio

Stephan Boecker, SAP Labs LLC, currently works as Solution Architect in the area of Manufacturing, with a focus on SAP xMII.

Introduction

Overview:

Using SAP xMII as extension of SAP XI at the plant site provides you with a consistent approach for system integration from ERP down to each plant site while keeping all the details of individual plant systems out of your overall integration scenario. The scenario would look as follows



Using SAP XMII integration capabilities at the plant site leaves you with the choice of whether to perform XML mappings in SAP xMII or in SAP XI. While your choice might mostly depend on your policy/preference or skill set available, it's worth noting the different approaches to interfaces/mappings in SAP XI and SAP xMII. SAP XI provides a clear distinction between interface/mapping development in the Integration Repository, configuration in the Integration Directory and its usage at runtime, a situation well-suited for standardized interfaces as well as 'centralized' development and roll out of manufacturing integration scenarios. SAP xMII uses a template approach, where each template serves as an example for a specific part of a business process, and is meant to be adapted to the individual needs at the plant site. Interfaces and XML mappings are usually only a part of the template. The approach is more suited for plant individual processes and a 'decentral' development.

'Generic Messaging' with SAP xMII

Using SAP xMII as the extension of SAP XI at the plant site leads to a reduction of service, mappings, and routings to be configured in SAP XI, since SAP xMII serves as counterpart. In certain situations, however, it might be necessary to know the original shop floor system involved, e.g to perform a specific mapping/routing. If XML mappings are done in SAP xMII as well, the information about the interface used by the shop floor system is not available, thereby reducing the information for routing even further. In order to help in this situation, XI content has been provided, which allows to wrap every XML document as 'inner child' of a generic XML container. This container can be filled during processing in SAP xMII with any information needed to perform detailed condition based routing in SAP XI. In addition two basic mappings are provided, which will 'pack' and 'unpack' an XML document inside this container. Sending information from SAP xMII to SAP XI using this approach would look like the following

```
<?xml version="1.0" encoding="UTF-8" ?>
- <ns0:GenericMessageSend xmlns:ns0="http://sap.com/xi/xmii">
    <OriginalParty />
    <OriginalService>ShopFloor</OriginalService>
    <OriginalInterface>Performance</OriginalInterface>
    <OriginalNamespace>shopns</OriginalNamespace>
    <IntendedReceiverParty>intrecPart</IntendedReceiverParty>
    <IntendedReceiverService>intrecServ</IntendedReceiverService>
    <IntendedReceiverInterface>intRec</IntendedReceiverInterface>
    <IntendedReceiverNamespace>intrecNS</IntendedReceiverNamespace>
    <TimeSent>2007-03-04T09:05:21Z</TimeSent>
  - <Payload>
      <Test />
    </Payload>
  </ns0:GenericMessageSend>
```

In this example a mapped XML document (containing only a root element <Test/>) has been entered as child named <Payload> into the container, which carries additional parameters with routing information about the sending shop floor system, its interface, the receiving system and its interface. The mapping program 'GenericSendMapping' would be used to 'unpack' the descendants of <Payload> which contains the already mapped XML document. Please note that you can fill in routing information at will in SAP xMII as long as you refer to it in SAP XI correctly. In the same way a document sent from XI to SAP xMII can be 'packed' via mapping program 'GenericReceiveMapping' into a container, where additional metadata about the sender, sender interface and receiver is provided:

```
<?xml version="1.0" encoding="utf-8" ?>
- <ns0:GenericMessageReceive xmlns:ns0="http://sap.com/xi/xmii">
    <SenderParty />
    <SenderService>TemplateSystem</SenderService>
    <SenderInterface>TemplateMessageSend</SenderInterface>
    <SenderNamespace>http://sap.com/xi/xmii</SenderNamespace>
    <ReceiverInterface>ReceiveGenericMessage</ReceiverInterface>
    <ReceiverNamespace>http://sap.com/xi/xmii</ReceiverNamespace>
    <TimeSent>2007-03-10T02:52:24Z</TimeSent>
  - <Payload>
    - <ns0:TemplateMessage xmlns:ns0="http://sap.com/xi/xmiitemp">
        <Value>TestMessage123</Value>
      </ns0:TemplateMessage>
    </Payload>
  </ns0:GenericMessageReceive>
```

In this case a service called 'TemplateSystem' did send an XML document, which his now the child of the <Payload> element. Mapping in SAP xMII to extract the original document is provided as well. The benefit of this approach is that if you do perform XML mappings in SAP xMII, the only thing needed in SAP XI is to use conditions for your routing. In the communication between SAP xMII and SAP XI only one (generic) interface is used, and only one (generic) mapping is needed in SAP XI. At this point the whole shop floor, including systems, interfaces, and mappings is transparent to SAP XI.

Asynchronous and Synchronous communication

The following paragraph is intended as a short overview about Messaging, and does not contain specifics about the package presented here. If you are familiar with those concepts from SAP XI, please read on with the next section.

Synchronous communication

Most developers are familiar with the concepts of synchronous communication, since it directly relates to the calling of a function or method in a program: You call something and get a reply back. A typical scenario is a user filling out search fields in an application. Upon pressing the 'Search' button, the underlying program executes the search, and presents the results on the screen. For programs, where a user expects immediate results containing data, this approach works well. However, in the area of Application-To-Application Integration, where one application sends data over the network to another one, it might not be the best choice. A typical example would go like this: An application sends data to an ERP system in order to post a goods receipt. The only 'response' the application requires, is some kind of 'OK-check', indicating that the document has been posted correctly, which is fundamentally different form the query response described before. The application itself is designed to execute this posting step in a synchronous way, so a user actually triggers the sending of data via pressing a 'Submit' button. If the posting succeeds, the user will receive a success message, other wise an error message will be displayed. In short, the application behaves like a single process application throwing an error. One problem with this design is, that applications, which send data over a network do not behave like applications running in a single process. For example the assumption, that a situation which does not result in a success message, is an error situation, does not need to be true, e.g if the posting did work, but the response never received the calling application, but instead timed out presenting the classical 'hourglass situation' to the user. From a user perspective the whole situation is pretty unpleasant: he has to trigger something a machine could do as well, has to wait for the posting and receives a response, that indicates an 'OK, I did it'. In case of an error, the user would typically try to overcome this error situation by pressing the 'Submit' button several times, hoping to 'finally get through', because otherwise he will lose the context of his session and has to start over again. On the ERP side this might result in multiple postings of a goods receipt, a situation that would not even be detected as error.

Asynchronous communication through XI

Asynchronous communication can overcome the problems with synchronous communication when it comes to sending data. In asynchronous communication, your application would typically not be driven by a user interface and deliver the data as message to a message broker (in our case XI), which would make sure, your data package gets to the intended receiver, similar to a service a post office would do with a physical package. The only part which would be synchronous in your own software, would be the part, where you hand over that data package to XI, or one of its adapters. While doing this, you would specify the 'Quality Of Service' you want that package delivered. XI or its adapters would make sure that

- Your data package either gets to the receiver or throws an alert in a monitored environment, for error correction or follow up
- Your data package does not get lost. This is the reason tat XI and its adapters persist messages as first step upon receiving
- A data package that was already send would not be sent again, a Quality Of service called 'Exactly Once'. This would avoid the double postings mentioned before
- If needed you can as well ensure the order of messages, e.g if you are sending several updates, you want to make sure the last one is in fact the last update. The corresponding quality Of Service is called 'Exactly Once In Order'

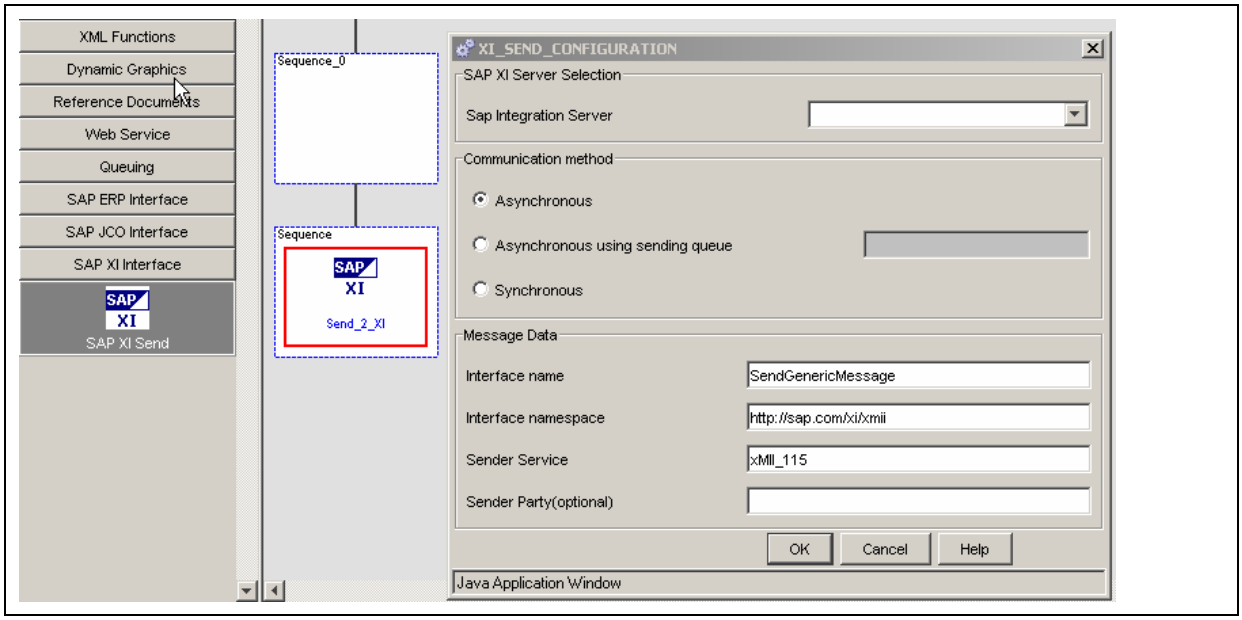
Since this hand over from you application to XI (or one of its adapters) only involves the persistence of the message on the XI side, you application will be more responsive. If it fails, you can reschedule the application without having to worry about 'double postings' since XI will reject a message already received. This identification of messages will be up to your application. Instead of execution a business transaction as

a single synchronous step, you thus build a message chain, where each member hands the message over to the next one. For this chain to work, you have to make sure that each member supports the needed 'Quality Of Service' and performs correct forwarding of messages. This is sometimes an issue in an environment, where the start of the chain (=the sender of the message) or the end of the chain (=receiver of message) do not provide adequate support. In addition, it is critical to have an efficient, end-to-end monitoring, to allow supervision of messages with respect to technical delivery, correct processing on the receiver side, as well as the ability to 'drill down' into problems, and provide error correction capabilities. To support this process, SAP manufacturing has created an own monitoring tool for 'Business Manufacturing Monitoring', which is especially aimed at the needs of manufacturing integration from ERP to shop floor, and will be published here shortly.

Components of the package

Action block to send messages to SAP XI

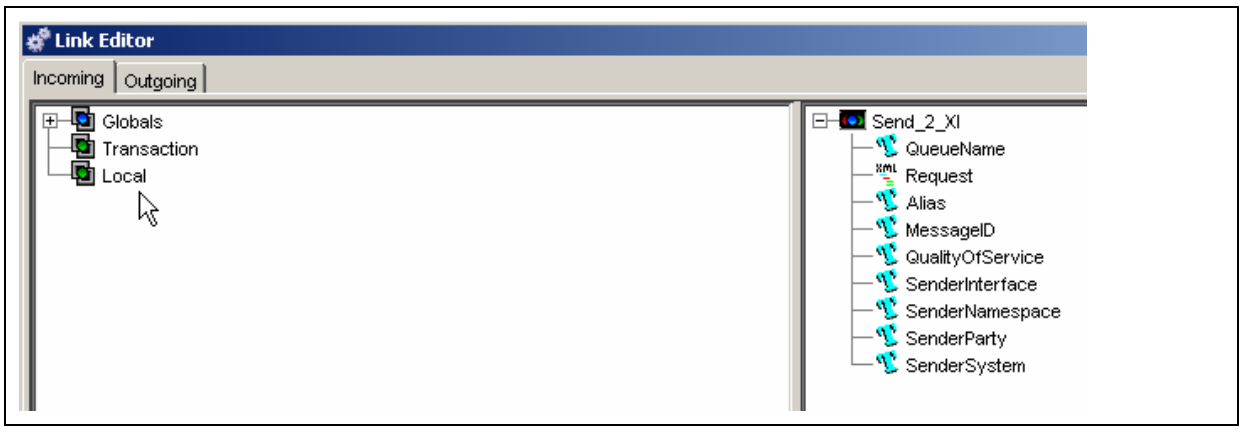
The package contains a java archive which will enable you to send XML data to SAP XI using an own action block as shown below:

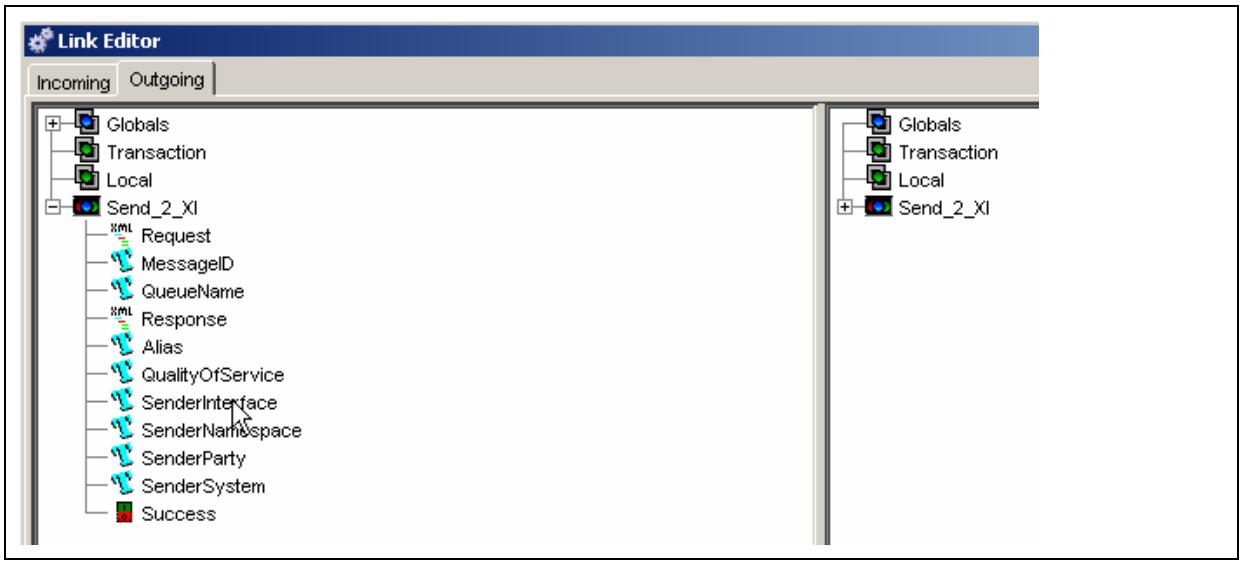


The action block requires a desired 'Quality Of Service' where

- Asynchronous corresponds to Exactly Once (EO)
- Asynchronous using sending queue corresponds to Exactly Once In Order (EOIO)
- Synchronous corresponds to Best Effort (BE)

In addition, you specify the 'Metadata' about your Message needed by SAP XI to perform a correct Receiver identification and Routing: Sender Service, Sender Interface, and Sender Interface Namespace. Specification of a sender party is optional. The XML data ('payload') you want to send, is specified in the Link editor as 'Request'. If you are sending data synchronously you would receive the Response XML document as 'Response', otherwise the 'Response' would be empty. In addition, the Link editor will provide the standard behavior of allowing you to overwrite the configuration settings at runtime:





Templates/pages for messaging support

In order to extend the quality of service form SAP XI to SAP xMII, a set of templates and corresponding web pages is provided. The templates/pages included provide

- Persistence of incoming/outgoing messages to ensure a 'Guaranteed delivery'
- Checking for duplicate messages to ensure a 'Quality of service' of Exactly Once
- Grouping of messages into queues to ensure a quality of service of 'Exactly Once In Order'
- Retry mechanism to allow message delivery if connections are down
- Error handling via administrator web pages. The error handling ranges from 'drill down' into the message, to manual delivery retry, and deletion of individual messages. For larger amounts of messages bulk operations like rescheduling of messages or queues is provided.

XI content

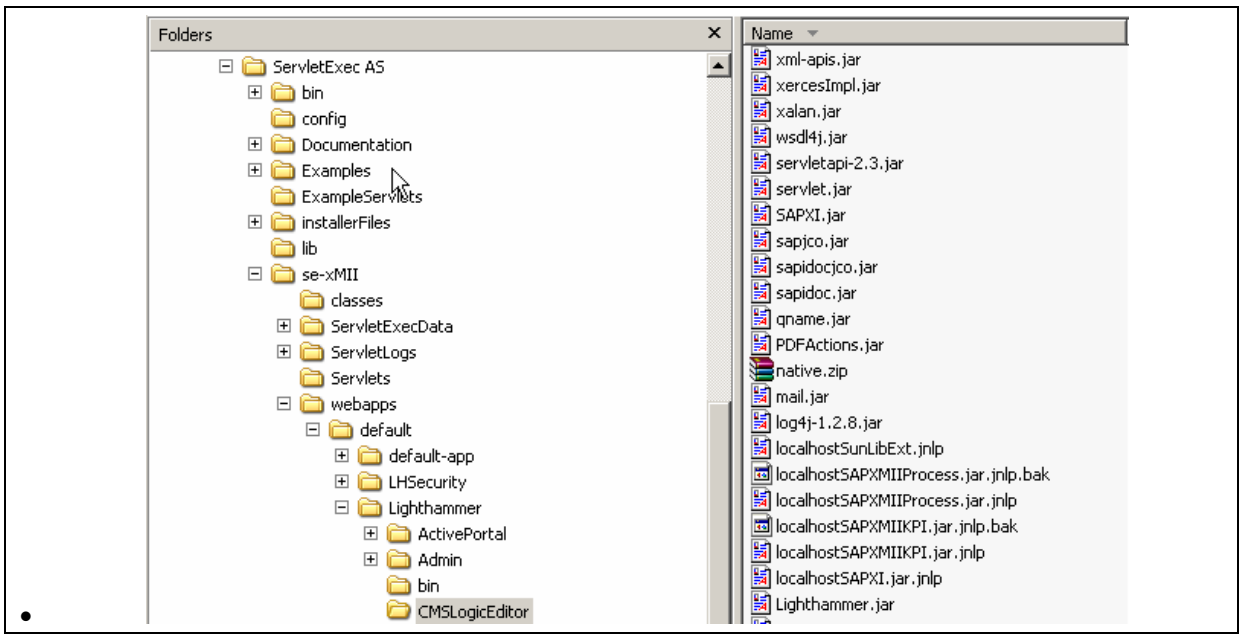
In order to help in the set up of messaging between SAP XI and SAP xMII, XI content has been included in the package. The repository part focuses on enabling 'Generic Messaging' between SAP XI and SAP xMII, and serves as a basis to test the scenario. The scenario is set up as exchange between a 'TemplateSystem' which serves as placeholder for any XI Business system or service, and SAP xMII. Interfaces and mappings are split between the 'TemplateMessages' and SAP xMII's 'GenericMessages', and further divided between asynchronous send/receive and synchronous communication between those two systems. The configuration content delivers templates on how to set up connections, sender agreements etc. between SAP xMII and SAP XI, as well as the 'TemplateSystem' which is accessible via File adapter.

Installation

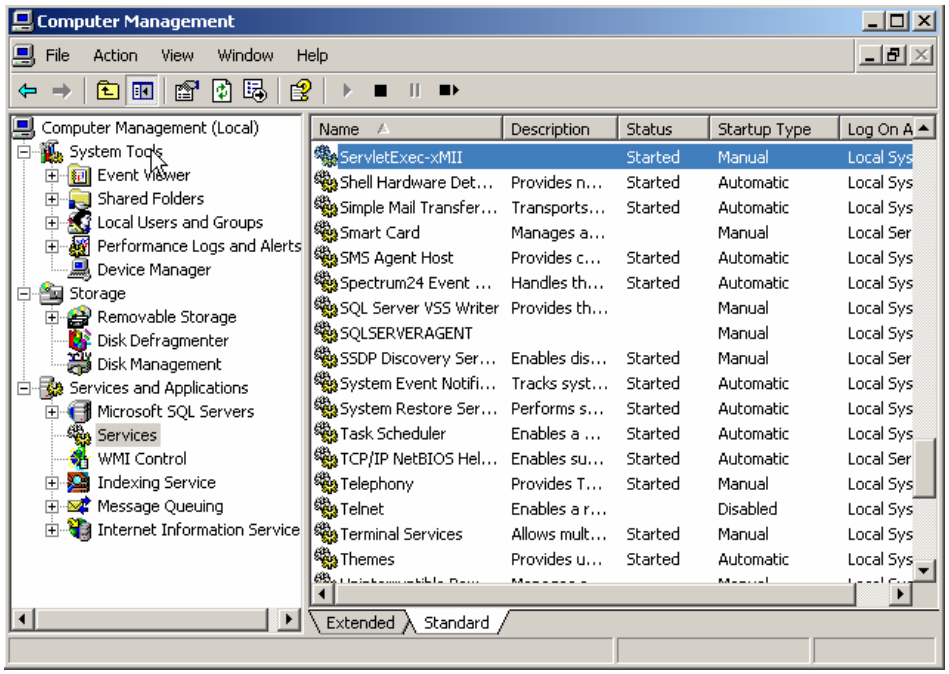
To perform installation steps take the archive *SAPXI.zip*, and extract it into a folder of your choice (which we will call the extraction folder) and extract it by double click

Java code for action block in SAP xMII 11.5

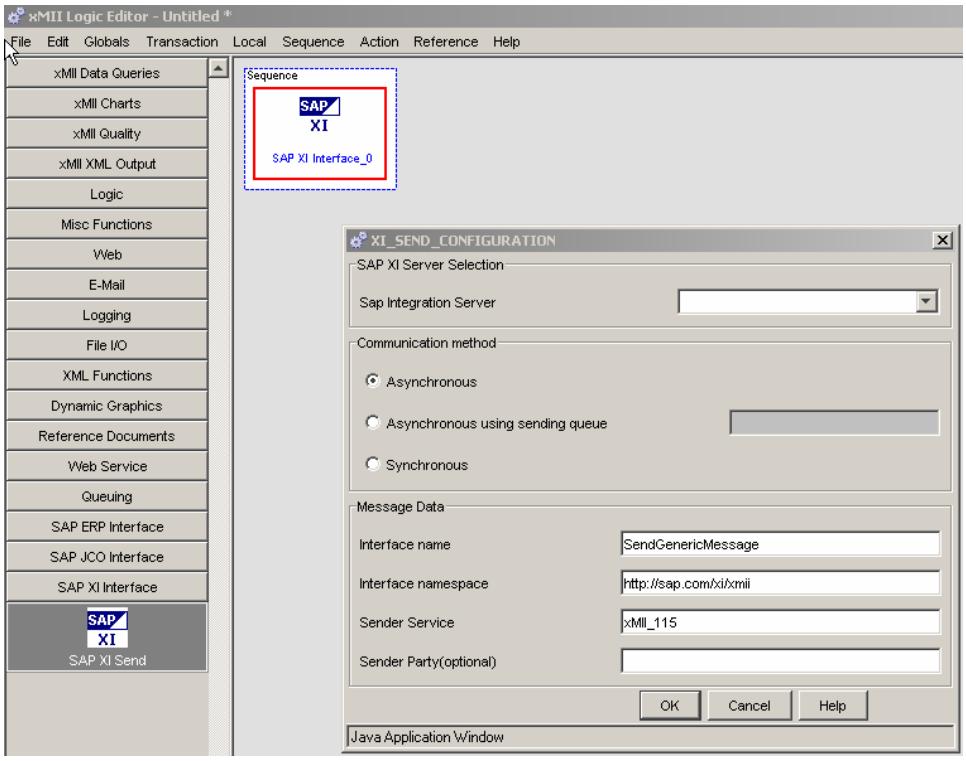
In order to send XML data to an XI Integration server an own 'action block' has been developed. In order to use this action block you have to put a .jar file in two places in the xMI server file system, and declare its use in SAP xMII. In order to install this file on your SAP xMII server, perform the following steps

- Locate the archive *SAPXI.jar* in the subfolder *software/java* of your extraction folder
- On your xMII server locate the directory called *ServletExec AS,* which should be on the drive where you installed SAP xMII.
- Navigate to the path *se-xMII/webapps/default/Lighthammer*
- Put the *SAPXI.jar* file in the subdirectory called *CMSLogicEditor.* This will make the action block available for the Logic editor in SAP xMII
- 

- Put the *SAPXI.jar* file in the subdirectory called *WEB-INF/lib*. This will make the action available for the runtime of SAP xMII
- Locate the xml file *XIComponentCatalog.xml* in the subfolder *software/java* of your extraction folder
- Put the file in the subdirectory *Lighthammer/Xacute/Components* of the drive where you installed xMII
- Restart the windows service *ServletExec-xMII*



- On your local PC where you are using the Logic editor navigate to the directory */Documents and Settings/<Your windows user name>/Application Data/Sun/Java/Deployment/javaws* and delete the subdirectory *cache*. This will ensure that the next time you are using the 'Logic editor' the .jar files are reloaded instead of using an outdated cached copy.
- Test it by launching the 'Logic editor' in SAP xMII. You should see an aditional Tab called *SAP XI Interface*. The action included there should provide a configuration dialog as depicted below.

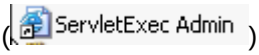


In order to install the help page associated with this action block, perform the following steps

- Locate the file *XISend.htm* in the subfolder *content/xMII/help* of your extraction folder
- Locate the web page directory (usually *C:/InetPub/*wwwroot) and navigate to the folder *LighthammerCMS/Help/Business_Logic_Services*.
- Copy the file to this location
- Locate the file XISend.gif in the subfolder *content/xMII/help/images* of your extraction folder
- Locate the web page directory (usually *C:/InetPub/*wwwroot) and navigate to the folder *LighthammerCMS/Help/images*.
- Copy the file to this location

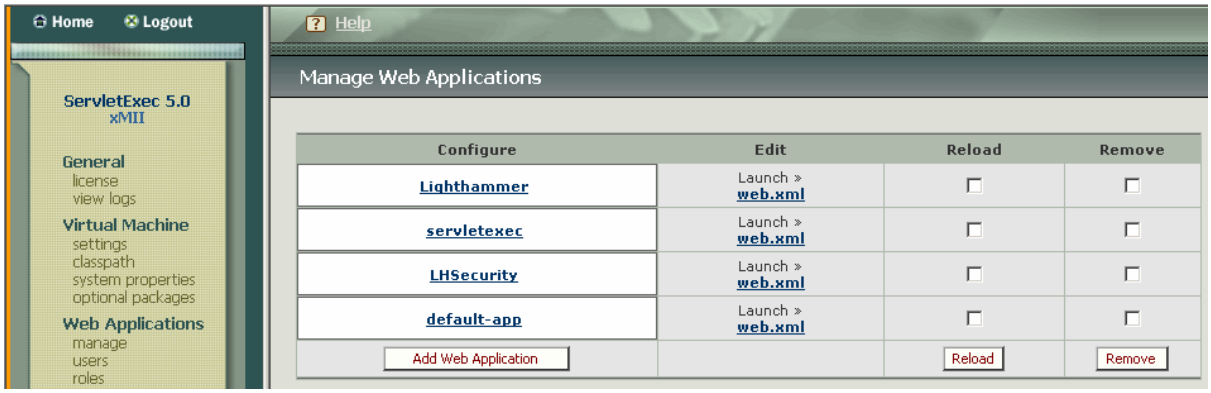
Java code for the servlet to be called by SAP XI

In addition to the action block the file *SAPXI.jar* contains as well a servlet which will be used by SAP XI to connect to SAP xMII. In order to use this servlet you have to declare its usage in ServletExec, which is the framework sitting between the Microsoft IIS server and the SAP xMII application. In order to declare the servlet. In order to do this perform the followng steps.

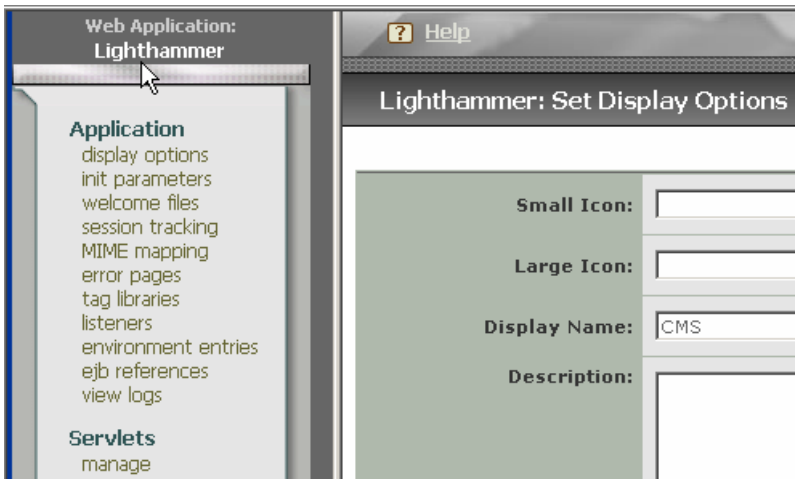
- On your xMII server locate the directory called *ServletExec AS,* which should be on the drive where you installed SAP xMII.
- Click on the link called *ServletExec Admin* (

). The *ServletExec Administration Login page* opens in your browser. Login with your servelt exec admin user and password you provided during installation of SAP xMII. The *ServeltExec Administration* page opens.

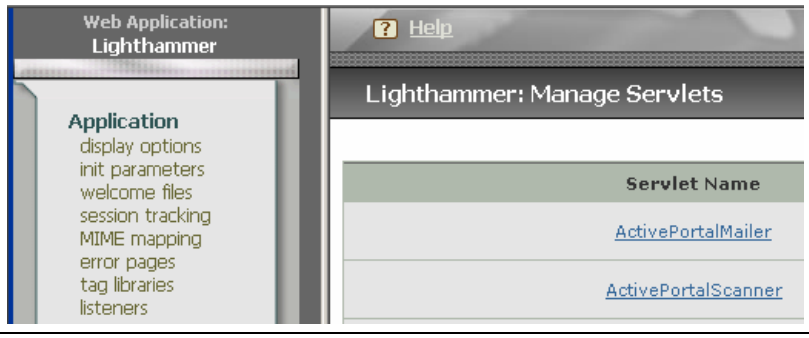
- On the left pane click the *manage* link in the category *Web Applications.* The *Manage Web Applications* screen appears in the right pane



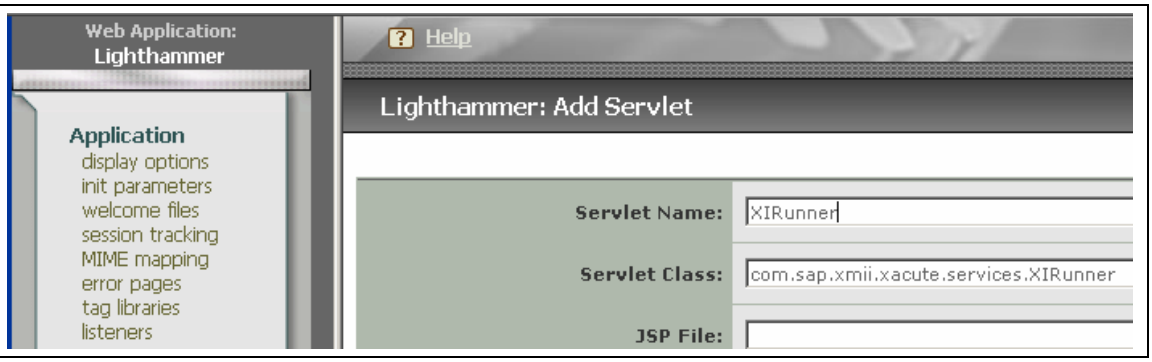
- Click the *web.xml* link underneath the column *Edit* in the row starting with *Lighthammer* in the first column. The *ServletExecAdmin* page opens in a new browser window.



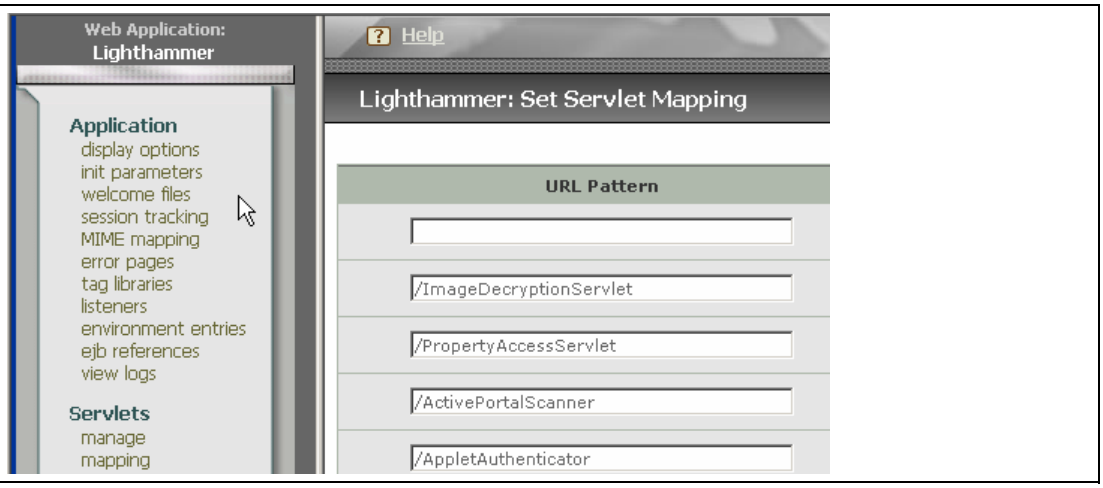
- On the left pane click the *manage* link in the category *Servlets.* The *Lighthammer Manage Servlets* screen appears in the right pane



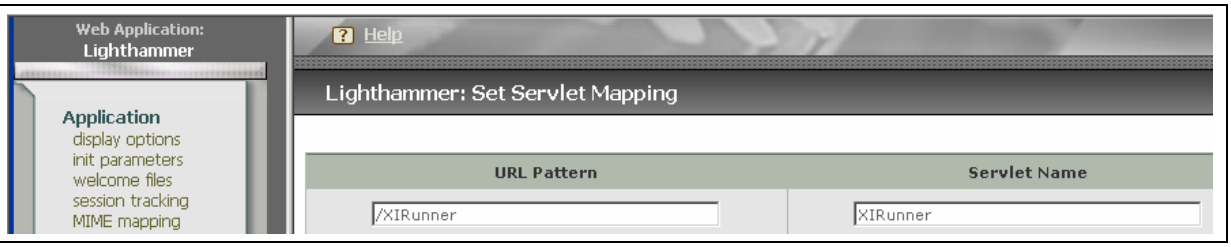
- Scoll down and press the *Add Servlet* button. The *Lighthammer Add Servlet* screen appears in the right pane.
- Enter **XIRunner** in the field *Servlet Name*
- *Enter* ***com.sap.xmii.xacute.services.XIRunner*** in the field *Servlet Class*

- 

  - Scroll down and press the *Submit* button. The *Lighthammer Manage Servlets* screen appears with a success message
  - On the left pane click the mapping link in the category Servlets. The *Lighthammer Set Servlet Mapping* screen appears in the right pane

- 

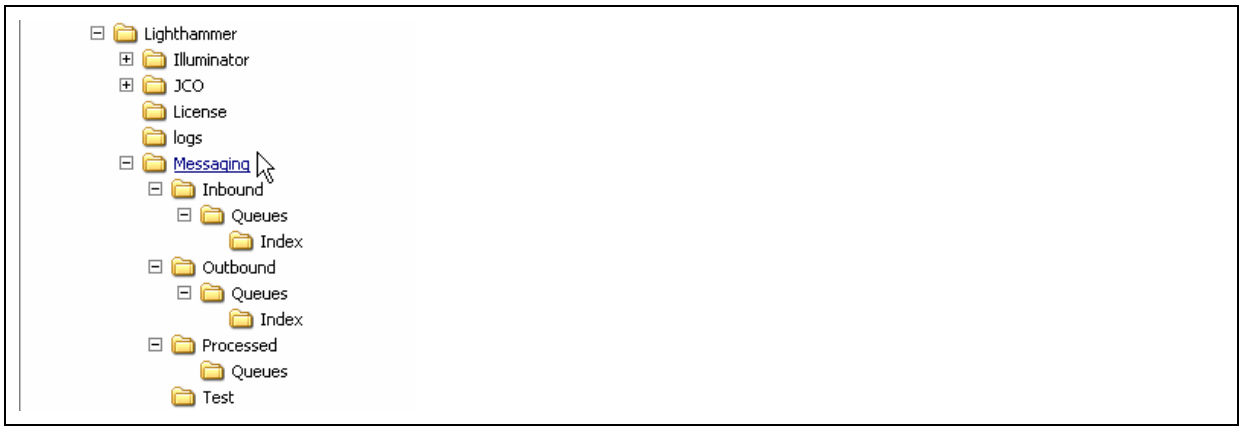
  - In the first line enter **/XIRunner** in the column *URL Pattern*
  - In the first line enter **XIRunner** in the column *Servlet Name*

- 

  - Scroll down and press the submit button. *Lighthammer Set Servlet Mapping* screen appears.
  - Scroll down the list to make sure your entry is existent
  - Restart the windows service *ServletExec-xMII* for your settings to take effect

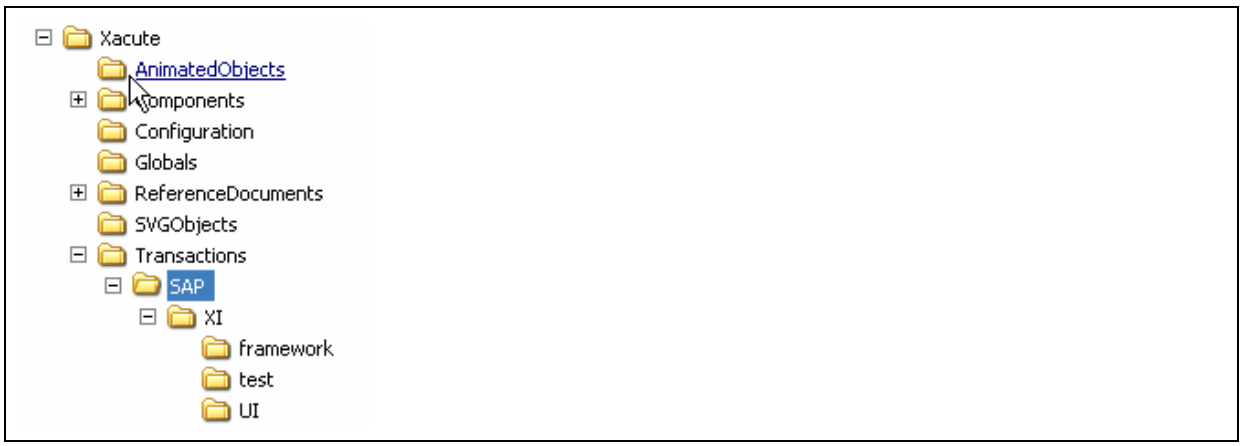
File system structure for messaging

In order to provide support for messaging from and to SAP XI you have to set up a directory called *Messaging* underneath your *Lighthammer* installation directory. The directory and its subdirectories should look exactly as follows



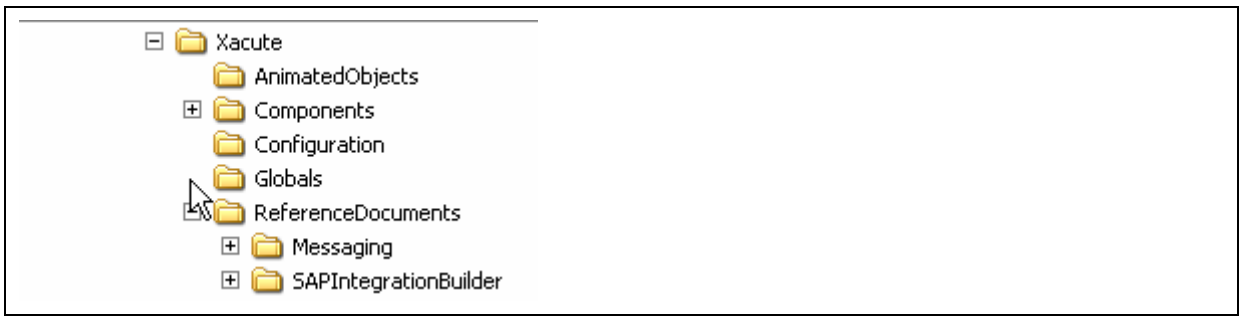
xMII transactions

A set of transactions has been developed for messaging support. You find these transactions in the *content/xmii/transactions* subfolder of your extraction directory. Copy the folder *SAP* and its subfolders as subfolder to the directory *<Drive>:/Lighthammer/Xacute/Transactions.* The result should look as follows



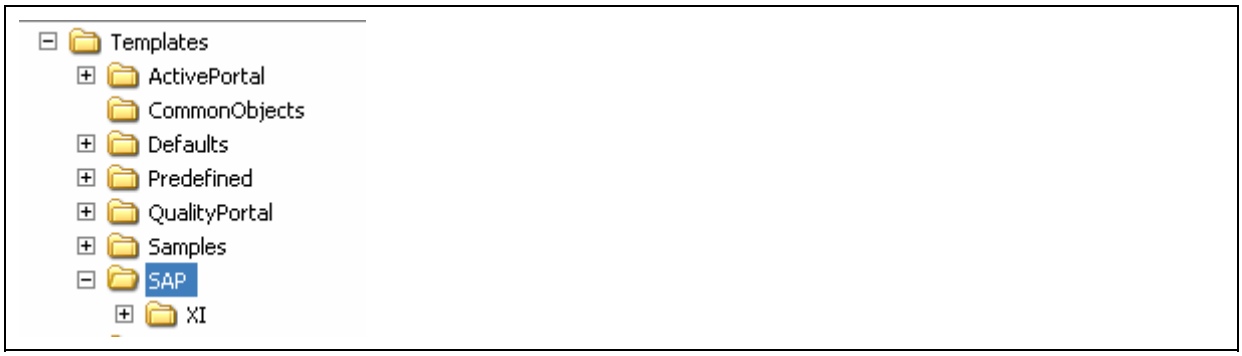
xMII Reference documents

A set of reference documents has been created for messaging support. You find these reference documents in the *content/xmii/ReferenceDocuments* subfolder of your extraction directory. Copy the content of this folder as subfolder to the directory *<Drive>:/Lighthammer/Xacute/ReferenceDocuments.* The result should look as follows



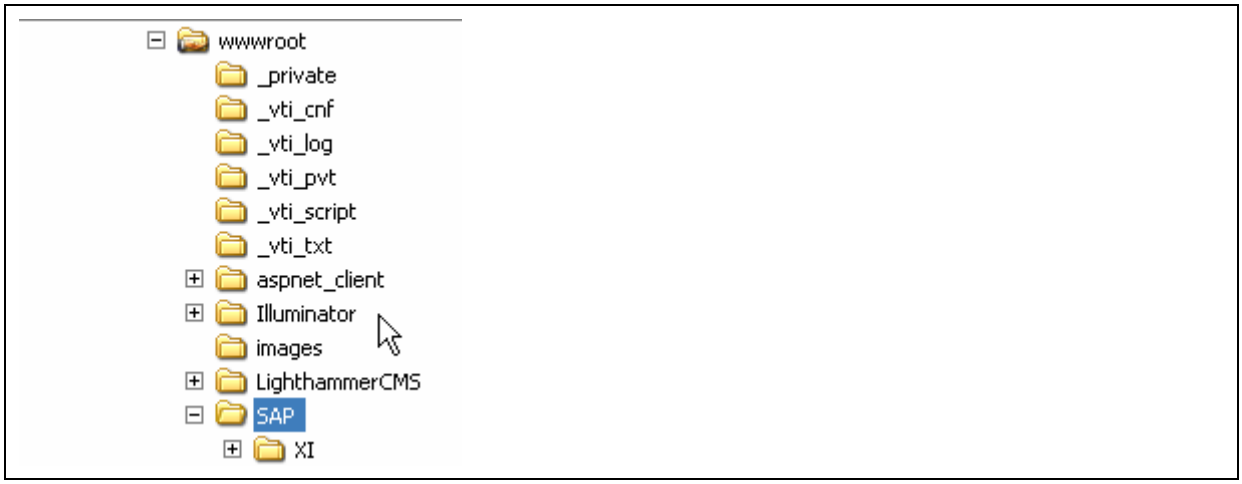
xMII templates

A set of templates has been developed for messaging support. You find these transactions in the *content/xmii/templates* subfolder of your extraction directory. Copy the folder *SAP* and its subfolders as subfolder to the directory *<Drive>:/Lighthammer/Illuminator/Templates* The result should look as follows



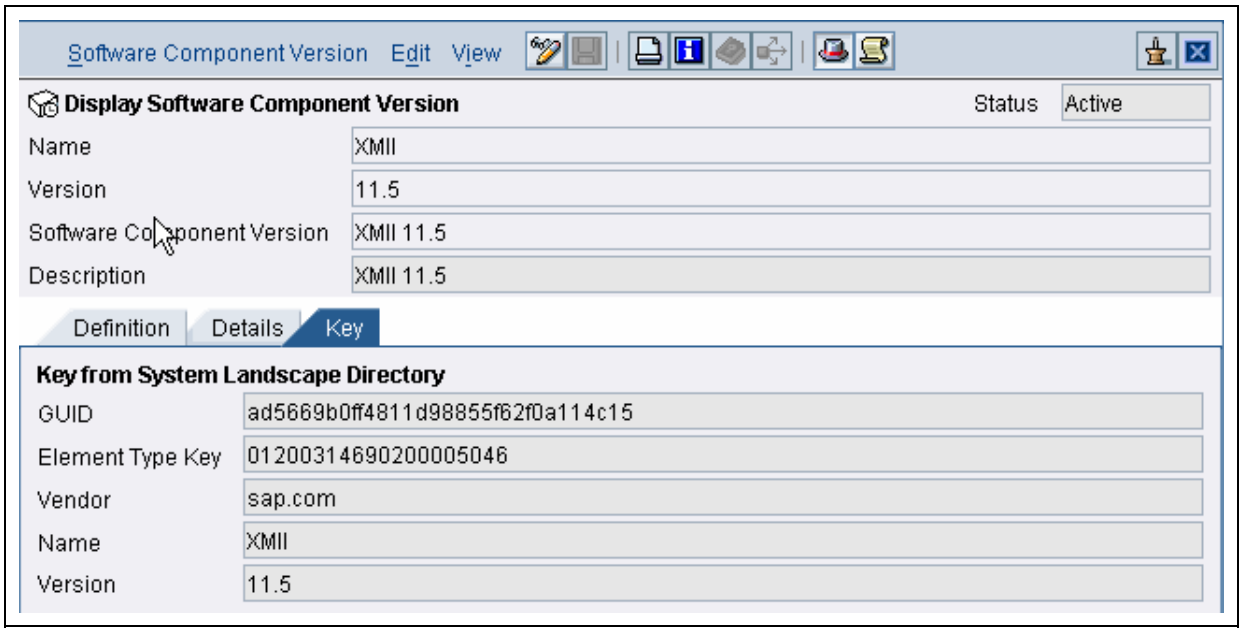
xMII web pages

A set of web pages has been developed for messaging support. You find these transactions in the *content/xmii/webpages* subfolder of your extraction directory. Copy the folder *SAP* and its subfolders as subfolder to your IIS web servers directory (which is usually *C:/Inetpub/wwwroot*), or a subdirectory you have chosen. The result should look as follows



XI repository content

XI repository content has been created to allow 'Generic Messaging' (see the paragraph abut generic Messaging below) The content can be found in the *content/xi/repository_server* subfolder of your extraction directory. The software component version used is as follows:



XI directory content

XI directory content has been created to allow 'Generic Messaging' (see the paragraph abut generic Messaging below) The content can be found in the *content/xi/directory_server* subfolder of your extraction directory.

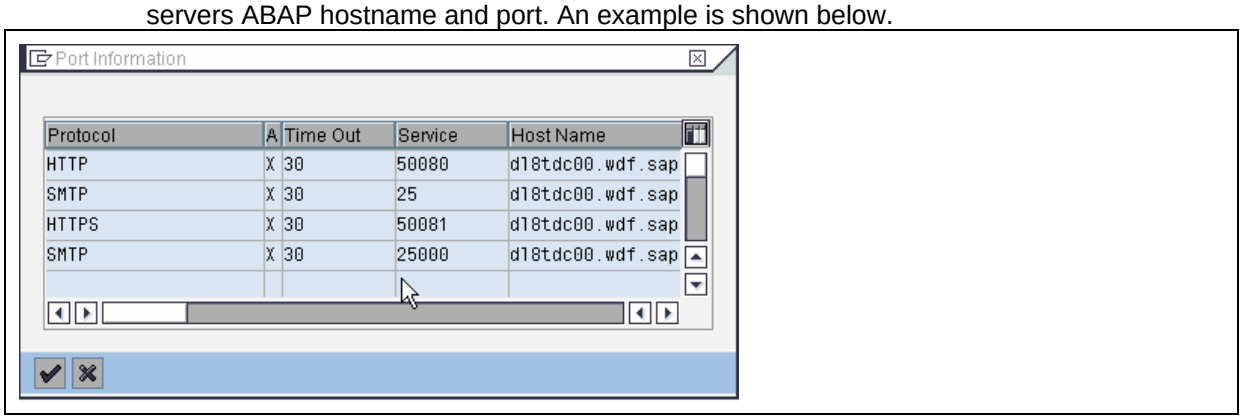
Configuration

The following paragraph describes the settings you have to perform to configure the scenario.

Get XI IS host name and port

xMII will use the HTTP adapter on the central Integration server ABAP instance, not the adapter framework. To get this hostname/port combination, perform the following steps

- On your central Integration server use transaction **sicf** to start the *Maintain services* application. Execute. The *Maintain Service* screen opens
- Press the *Information on Port and Host* icon ( ) to get a popup specifying the Integration servers ABAP hostname and port. An example is shown below.



Define XI Integration server in SAP xMII

Perform the following steps

- In your SAP xMII Menu open the *SAP server Configuration* in the category *Data Services*.
- Select the *SAP_WAS* template in the left pane and press the *Copy* button. The right pane opens for editing.
- Enter the SAP XI IS server's hostname identified in the previous step in the field *Server*
- Enter the SAP XI IS server's port in the field *Port*. Note that XI IS used different ports for Http and Https, as shown in the previous section
- Enter a logon user in the field *UserName*. The default for this user in an XI environment would be the user named **XIAPPLUSER**.
- Enter the users password in the field *Password*
- Enter the user logon language in the field *Language.*If you do not enter a value the default language of English is applied
- Enter the XI IS logon client in the field *Client*
- Enter a response time out in seconds in the field *ResponseTimeout.* If you do not enter a value a default of 60 seconds is applied
- Mark the *SSL* check box if you want to use Https. Please note that this will require the correct Https port to be entered in the field *Port*
- Save your settings by pressing the *Save* button.

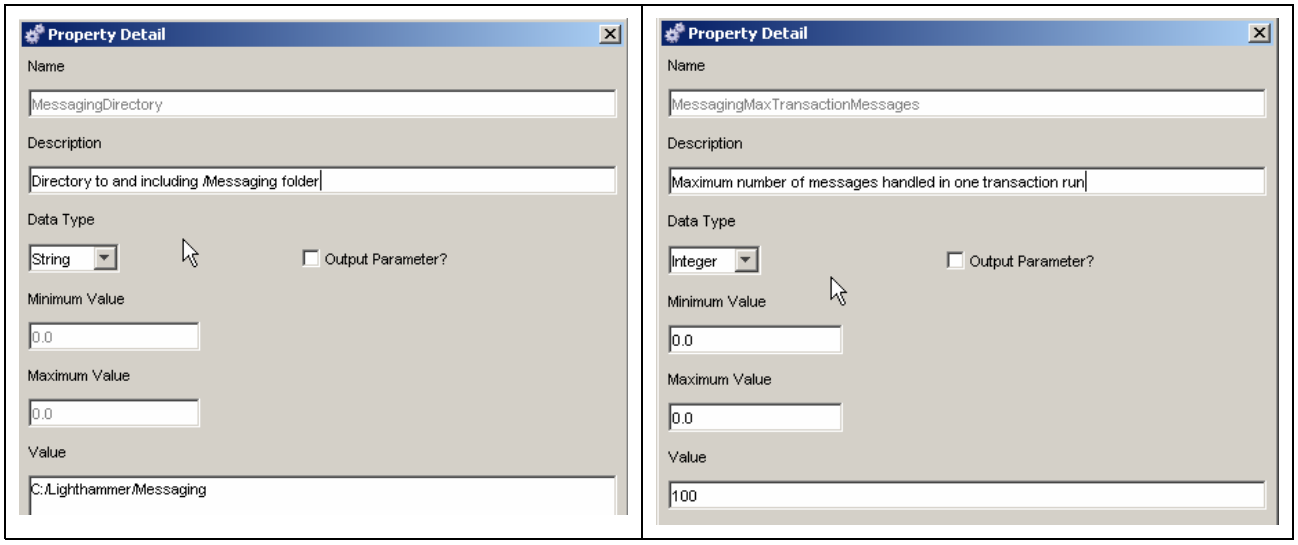
Set up Global Properties in xMII for messaging

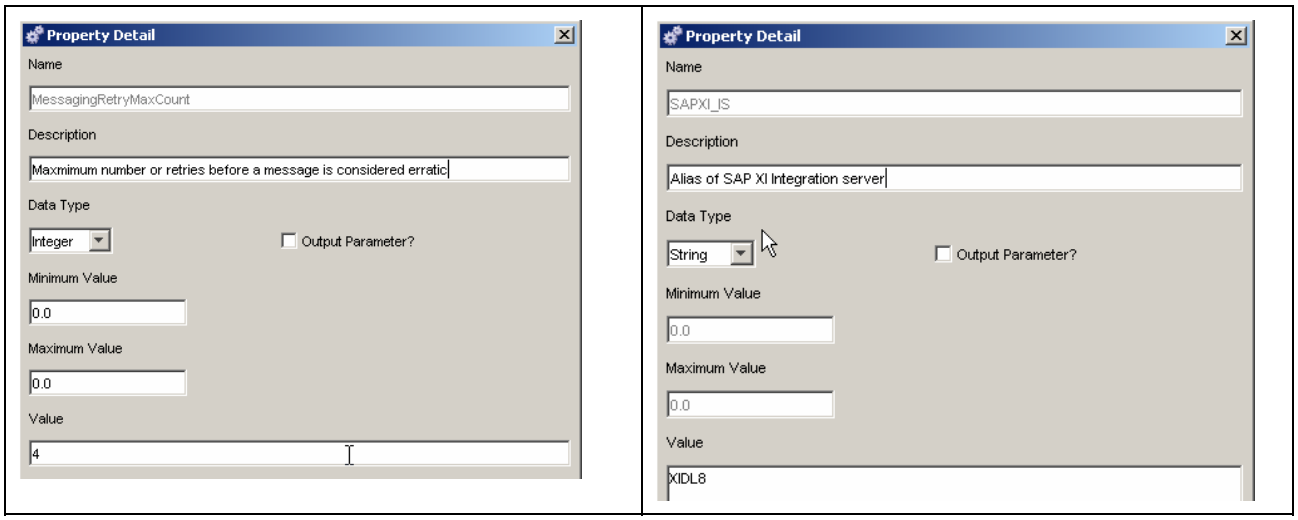
Perform the following steps to set up the global properties for Messaging in SAP xMII

1. In your SAP xMII Menu open the *Logic Editor* in the category *Business Logic services*. The *xMII Logic Editor* Window opens.
2. In the menu bar choose *Globals->Properteis.* The *Global Properties* popup appears.
3. Press the *Add* button to add an entry
4. Enter the property *MessagingDirectory* in the field *Name*.
5. In the field *Value* enter the path to and including the directory you created in section *File system structure for messaging,*
6. Enter a descriptive text in the field *Description*
7. Press the *OK* button to save your entry

- Repeat the steps 3 until 7 for the property *MessagingMaxTransactionMessages* with a value of **100**. This property limits the number of messages processed in one transaction run to a maximum of 100 in order to avoid a timeout in transaction execution for large message queues. You can later adjust this number as long as you are making sure that the number is not so large that either transactions will run into a timeout or scheduled transactions are starting to run in overlap.
- Repeat the steps 3 until 7 for the property *MessagingRetryMaxCount* with a value of **4**. This property defines the number of attempts made to deliver a message before the message is considered 'executed with error'
- Repeat the steps 3 until 7 for the property *SAPXI_IS*. In the field *value* enter the alias of the SAP XI IS you specified in section *Define XI Integration server in SAP xMII*. This property is used by the framework transactions to address the correct XI IS.

An example would look like the following:





Schedule transactions for messaging processing

Use the Schedule Editor in the category to schedule the transaction handling messaging.

Schedule the following transactions (recommend is every 30 seconds to every 2 minutes, depending on your needs and the processing speed of messages until the next scheduled run starts)

- Schedule the transaction *SAP/XI/framework/ProcessPersistedInboundMessages*. This transaction takes persisted messages received from SAP XI and executes your individual transaction. If execution succeeds, the messages will be stored as "Processed'. Otherwise they will be executed repeatedly until the maximum retry count is reached. At that point a message is considered executed with errors and will not be rescheduled. Use the administrative web pages to supervise those error cases
- Schedule the transaction *SAP/XI/framework/ProcessPersistedInboundQueueMessages.* This transaction acts identically to the previous one with the exception of handling queued messages received from SAP XI.
- Schedule the transaction *SAP/XI/framework/ProcessPersistedOutboundMessages*. This transaction takes persisted messages received from the shop floor or created by your own transaction in SAP xMII and sends the data to SAP XI. The behaviour in terms of retries and erros is identical to the corresponding inbound transaction
- Schedule the transaction *SAP/XI/framework/ProcessPersistedOutboundQueueMessages.* This transaction acts identically to the previous one with the exception of handling queued messages to be sent to SAP XI.
- Schedule the transaction *SAP/XI/framework/CleanUpMessageFiles* which will delete messages files processed successfully. The recommended schedule is once per day. In the input parameter *TimeoutInHours* enter the number of hours, after which a message is considered outdated and can safely be deleted (default would be 24). Please not that you should not schedule this transaction to close to the receiving/processing of messages, since the directory with processed messages is need to identify incoming messages as 'doublets' for a Quality of service of 'Exactly Once'

Set up connection from XI to xMII

In the configuration content provided in the package SAP xMII is specified as Business Service with an HTTP endpoint. The same steps described here apply if you prefer the use of Business systems.

In order to point the communication channel to your SAP xMII server perform the following steps

- Open the scenario *xMIIGenericMessaging*

- Navigate to the Business service *xMII_115* and underneath that node, navigate to the communication channel *HTTP*. The *Display Communication Channel* screen opens in the right pane.
- Press the *Switch between Display and Edit* mode icon.
- In the field *Service Number*, enter the port of the xMII application
- In the field *Target Host*, enter the hostname of the xMII server
- In the field *Path* enter the path to the xMII servlet handling XI requests, which is **/Lighthammer/XIRunner?InputParameter=Payload&OutputParameter=ResponseDocument&Transaction=SAP/XI/framework/ReceiveXIMessage&XacuteLoginName=<xMIIUser>&XacuteLoginPassword=<xMIIPassword>**, where <xMIIUser> and <xMIIPassword> have to be replaced with a valid xMII user name and password. Please note that issues as Single-Sign-On are addressed in the security section of this article.
- In the dropdown box *Authentication Type* choose the entry *Anonymous Logon*
- In the field *Content Type* enter **text/xml**
- In the section *Additional Query String Attribute* make sure that all checkboxes are marked.
- In the field XML Code enter **UTF-8**
- Save your settings by pressing the *Save* button.

Process flow

The following chapter will give you an overview of the process flow and how you can use the transaction callbacks built into the package

Message flow XI->xMII

Messages arriving from SAP XI adress the XIRunner servlet, which in turn will call the transaction

SAP/XI/framework/ReceiveXIMessage

Synchronous messages

If the message is synchronous the transaction *SAP/XI/ProcessSynchronousIncomingXIMessage* will be called which will forward to the transaction *SAP/XI/YourSyncIncomingXIMessageTransaction.* You should use this transaction as a starting point to process the request. Since it is a synchronous scenario you have to assign a value to the *OutputXML* output parameter specifying the response XML document. This document will be sent back as payload to the calling SAP XI IS.

Asynchronous messages

If the message is asynchronous the transaction *SAP/XI/ProcessIncomingXIMessage* will be called. which in turn will call the transaction *SAP/XI/YourAsyncIncomingXIMessageTransaction*. In this transaction you can overwrite 'Message Metada', which will be assigned to the message. Those metadata will later serve in xMII to process the message correctly. However, those metadata are purely optional and partially already filled by the incoming XI request (like receiver interface and namespace, for example). The only parameter you might want to change is the name of the xMII inbound queue, if the required quality of service is 'Exactly Once In Order'. If you do not specify an own queue for this quality of service, xMII will apply as default queue name the combined names of receiver interface and namespace. Please note that this transaction it **not** yet the place to perform any complex business logic, since it happens still in the request/response cycle form XI to xMII. After calling *YourAsyncIncomingXIMessageTransaction* control is forwarded to the persistence transaction, which will check for duplicated message IDs and persist the message If errors are detected during this process, an http error response is sent back to SAP XI IS, otherwise an Http 200 code is returned.

Internal processing and delivery to the shop floor

The 'Inbound' transactions you scheduled in section *Schedule transactions for messaging* are used to handle further processing to the shop floor or inside of SAP xMII. The processing sequence is the same for queued and not queued messages with respect to call backs. After reading the message ad checking on error conditions, transaction *SAP/XI/framework/ProcessPersistedGeneralMessage* is called, which in turn hands control over to transaction *SAP/XI/ProcessPersistedIncomingXIMessage* which in turn will call the transaction *SAP/XI/YourAsyncIncomingPersistedMessageTransaction.* This is the transaction you should use as starting point to develop your own business logic. The already existing action blocks perform checks if the incoming message is of the 'Generic' message type specified in the XI content, and extracts the inner message stored as payload inside this Generic message. Further processing in our implementation is to simply store the message in another folder. You can replace this piece with any transaction you want executed at this point in time. If the transaction does not terminate with error, the message is considered 'processed' and will be moved to the *Processed* folder. Otherwise processing will start again with the next run of the scheduled transaction, until the maximum processing count has been reached. At this point the message will be marked as erroneous. Please read on in the section about error handling for more details on how to handle error messages.

Messages from shop floor to SAP xMII

Outbound processing is initiated by calling transaction *SAP/XI/ReceiveOutgoingMessage.* This transaction takes the XML document to be forwarded as *Payload*. In addition it will allow you to set 'Metadata' about this message. This metadata falls into 3 groups

a) Metadata about the message itself like *QueueName, MessageID*, and *QualityOfService*

b) Metadata about the 'sending system' like *OriginalSenderInterface*, *OriginalSenderNamespace* etc.

c) Metadata about the 'receiving system' like I*ntendedReceiverInterface*, *IntendedReceiverNamespace* etc.

Please note the following

1. Metadata in a) and b) are similar to parameter names which should be familiar if you know the concepts in XI. However they are not intended to correspond to real interface and business systems in XI. They are a set of variables allowing you to specify any string to identify to SAP xMII (and later XI), who the sender, receiver, and what their respective interfaces are. As an example you could have a *OriginalSenderService* of 'Shopfloor' and *OriginalSenderInterface* of 'Performance' as long as you know how to perform mapping in SAP xMII (or XI) and - if needed- trigger the correct routing in XI based on that information coming form the plant systems
2. The only obligatory fields you need to fill are the *Payload*, *QualityOfService*, *OriginalSenderInterface*, and *OriginalSenderNamespace*, in order to allow SAP XMII to identify who the sender is, what is sent and what Quality of service is required
3. If you do not specify a *QueueName*, but specified *QualityOfService* to be EOIO ('Exactly Once In Order', SAP xMII will use the information in *OriginalSenderInterface* and *OriginalSenderNamespace* to create a queue name for you.
4. If you do not specify a *MessageID*, a new unique *MessageID* will be created. If you want a Quality of service of ExactlyOnce, you therefore should create your own *MessageID*, so SAP xMII can check for duplicates upon message arrival.

The transaction called will trigger the persistence of this outbound message. It will fill an Output parameter *ReturnCode* with the value 0, if persistence was successful. If the *ReturnCode* is not 0 or if you receive an output document not containing an *ReturnCode*, the transaction *SAP/XI/ReceiveOutgoingMessage* has to be called again. The persistence includes the XML document and a document containing the 'Metadata', which ends with a *.met*

Transaction *SAP/XI/test/TestShopFloorSend* contains an example on how to trigger outbound processing.

Message flow SAP xMII ->SAP XI

Synchronous messages

The sending of synchronous messages is performed by using the action block 'SAP XI Send'. Specify the sender system/service, sender interface, interface namespace, and optionally sender party in the configuration. Mark the radiobutton named *Synchrounous* in the group *Communication method.* After executing the action block, the synchronous response XML document can be found in the link editor as *Response*.

Asynchronous messages

The sending of messages to SAP XI is triggered by the scheduled transactions *SAP/XI/framework/ProcessPersistedOutboundMessages* and *SAP/XI/framework/ProcessPersistedOutboundQueue Messages.* After checking on error conditions on the file the transaction *SAP/XI/framework/ProcessPersistedGeneralMessage* is called, which hands control over

to transaction *SAP/XI/framework/SendMessage2XI*. This transaction will call transaction *SAP/XI/ProcessPersistedOutgoingXIMessage* and pass the payload XML structure as well as the metadata XML structure. In this transaction the actual assembly of the payload for SAP XI and the specification of sender service, interface, and namespace takes place, since shop floor systems usually do not provide this information. The entry point for setting those parameters is in transaction *SAP/XI/YourAsyncOutgoingXIMessageTransaction*, which is called first. Please check the actual implementation on how to use the metadata XML input provided to set the corresponding parameters for SAP XI, an example based on the 'TemplateSystem' provided in the XI content. Please note as well that if you do not provide those parameters in transaction *SAP/XI/YourAsyncOutgoingXIMessageTransaction,* the calling transaction *SAP/XI/ProcessPersistedOutgoingXIMessage* will proceed assuming you want to use GenericMessaging with a service name of *xMII_115.* Finally, transaction *SAP/XI/framework/SendMessage2XI* will send the message to SAP XI with the Quality Of service specified before. If the transaction does not terminate with error, the message is considered 'processed' and will be moved to the *Processed* folder. Otherwise processing will start again with the next run of the scheduled transaction, until the maximum processing count has been reached. At this point the message will be marked as erroneous. Please read on in the section about error handling for more details on how to handle error messages.

Error handling

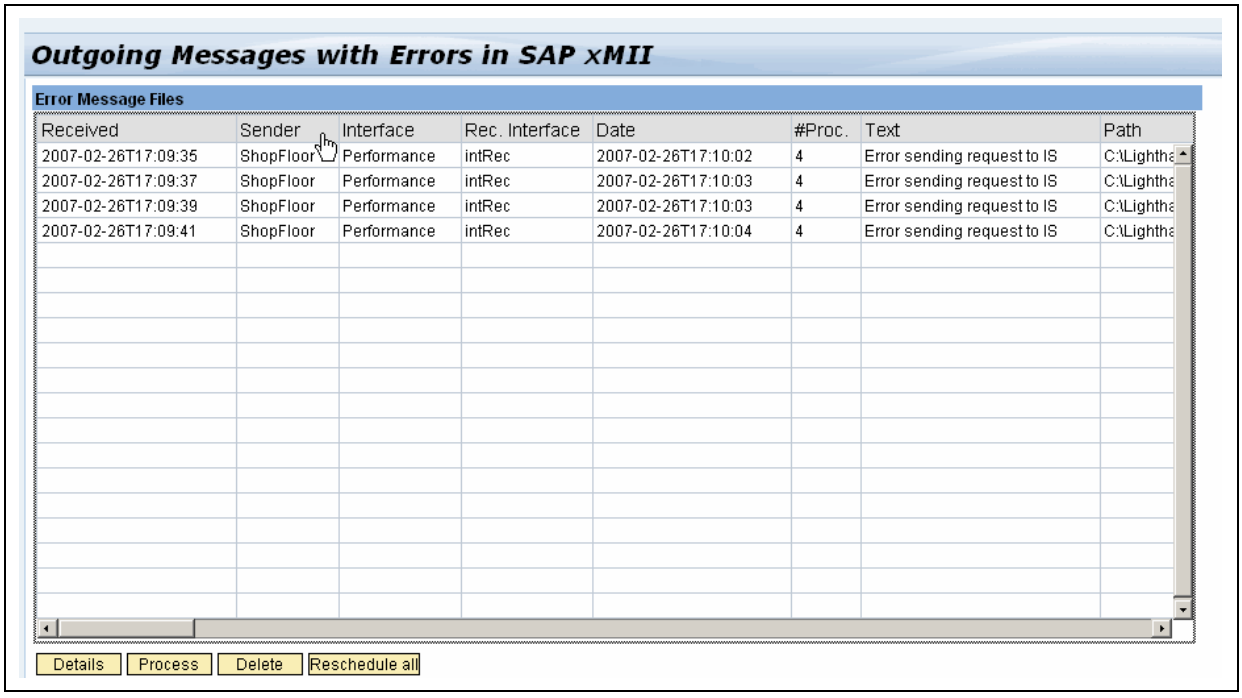
Messages which can not be sent to SAP XI or which can not be processed internally, will be re-processed until the counter as specified in the Global Variable *MessagingRetryMaxCount* is reached. At this point, as already mentioned, messages receive an error flag, which is indicated by

- A lock document ending with *.lck*
- An error text document containing the last error message ending with *.err*

This marking happens in transaction *SAP/XI/framework/StoreMessageAsError*, which is called by transaction *SAP/XI/framework/ProcessPersistedGeneralMessage.* If you want to enhance error handling by e.g sending e-mail alerts, you should do this in *SAP/XI/framework/StoreMessageAsError*, . Messages marked as error will not be processed during subsequent scheduled runs. It is up to the user to correct the problem, delete the message or simply reschedule the message for execution. To help with this process, web pages have been provided, where an administrator can perform those tasks. The following sections walk you through these pages

Inbound/Outbound Messages (not queued)

Message marked as error can be view via link *http://<server>:<port>/SAP/XI/Messages/LockedMessages.irpt?IsInbound=1* for inbound messages from SAP XI, which could not be processed internally and *http://<server>:<port>/SAP/XI/Messages/LockedMessages.irpt?IsInbound=0* for outbound messages to be sent to SAP XI. An example would look like the following:



You select a message by marking the corresponding line. For each message you can perform the following actions

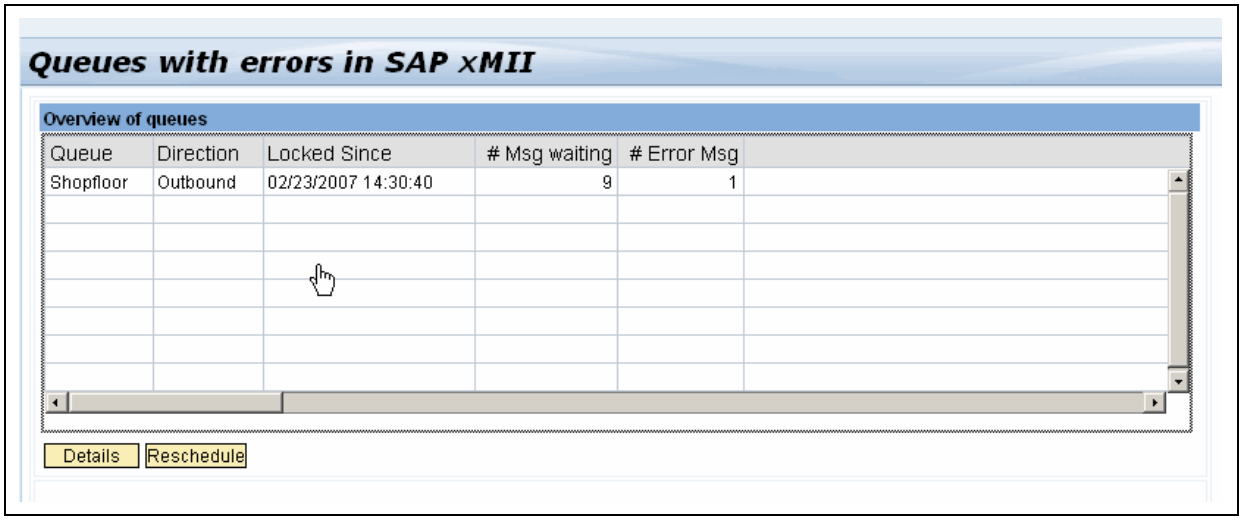
- View the message by pressing the button *Details*
- Trigger processing of the message by pressing the button *Process*. If processing succeeds the message should disappear form the list, otherwise the error counter (*#Proc*) should increase to indicate an unsuccessful processing.

- Correct the message via the information given in column *Path*
- Delete the message permanently by pressing the button *Delete*

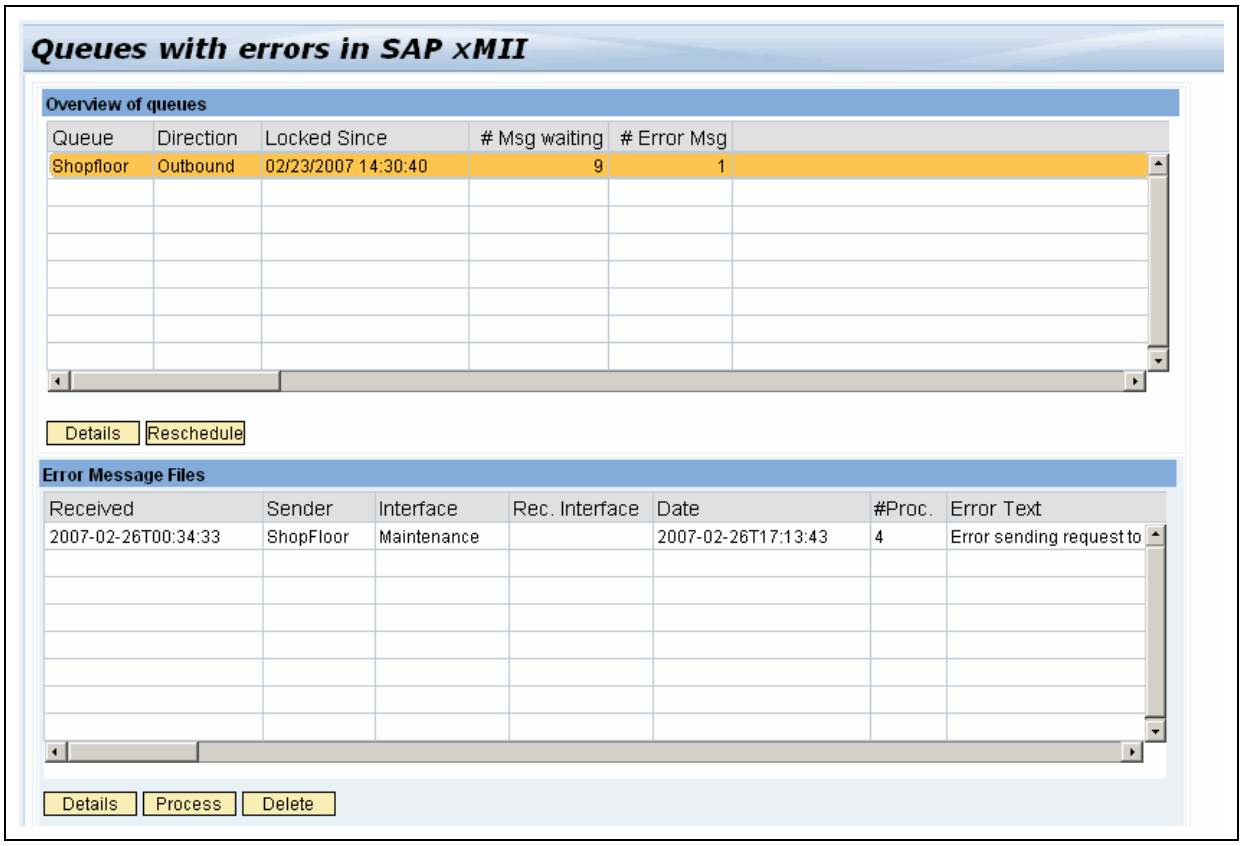
In addition you can reschedule a bulk of message for processing by pressing the button *Reschedule all* without marking any line. In this case all error and lock files will be deleted and the message will be processed with the next scheduled run. Please note, that the maximum number of messages, which can be unlocked in a single transaction run is restricted to three times, in order to avoid transaction timeout. If you have a large amount of messages, you therefore might have to press the *Reschedule* button several times in order to unlock all messages. The transaction is *SAP/XI/framework/UnLockErrorMessages*

Queued Messages

Queued processing behaves differently than processing of 'regular' messages, since a message marked as erroneous will prevent the further processing of not only the message itself, but all messages which were received at a later point in time, resulting in messages piling up in the queue 'behind' the message containing errors. You can see queues, which are 'piled up' via the link *http://<server>:<port>/SAP/XI/Queues/LockedQueues.html.* An example would look like this



Each entry represents a queue including the number of messages waiting and the number of error messages. By marking a line (and optionally Pressing the *Details* button) you can see the individual messages:



The functions for individual messages are identical to the ones mentioned in the previous paragraph. However, in order to guarantee the order of the queue, you can not process individual messages in a queue. You can either correct the message with errors (scroll to the right to the path to the message document) or delete the message permanently. Afterwards you can reschedule the whole queue via pressing the *Reschedule* button above. The queue will be processed with the next scheduled run. Please note, that the maximum number of messages, which can be unlocked in a single transaction run is restricted to three times, in order to avoid transaction timeout. If you have a large amount of messages, you therefore might have to press the *Reschedule* button several times in order to unlock all messages. The transaction is *SAP/XI/framework/UnLockQueue*

Please note as well that if you reschedule the queue without correcting the message containing the error your queue will begin piling up again, since the oldest message will be again the first to be executed.

Security Settings

Logon

Logon to SAP XI

The login performed in SAP XI from SAP xMII is performed via Basic Authentication as default. Please verify that the logon procedure for the service /default_host/sap/xi/adapter_plain supports this type of logon, e.g via logon procedure *SAP Standard*. Another logon method which is not activated per default is form based logon via the URL query fields sap-user etc. If you want to use this logon procedure uncomment the block in the method g*etServerURL()* of java class *SAPXIConfiguration.java* , and repack the SAPXI.jar archive.

Logon to SAP xMII

The login to SAP xMII is perfomed as form based logon via the URL parameter *XacuteLoginName=<xMIIUser>&XacuteLoginPassword=<xMIIPassword>*, which you specified in the HTTP adapter for the service *xMII_115.* You can omit these two parameters if ypou wnat to perform Single-Sign-On (SSO) from the XI Integration server to SAP xMII. The process on how to set up this connection is documented on SDN.

SSL

SSL from SAP xMII to SAP XI

The package supports SSL via https URL's from SAP xMII to SAP XI, if you checked the box *SSL* in the *SAPServerConfiguration* in SAP xMII. Obviously as a precondition you have to configure your SAP Web Application server running the XI Integration server to support server SSL. The code in SAP xMII uses the default JSSE implementation with one exception, when the client validates the servers certificate sent during handshake. The reason for this is that scenarios between SAP XI and SAP xMII are pure intra company scenarios, where the SAP Web Application serve does not necessarily need a signed certificate in order to perform encryption. The default JSSE implementation would always require a signed certificate. The actual code therefore uses an own Trustmanager in class *XITrustmanager.java* , which will evaluate if the certificate sent from the XI Integration server is still valid with respect to date, but not if there is a valid certificate chain in place. If you want to switch back to the default implementation, which means you do have a valid certificate in place for the XI Integration server, please use the comment indicated in method *checkServerTrusted(X509Certificate[] certificates,String authType)* to use the JSSE default. Additionally, you can provide specific Sender agreements in XI configuration to enforce https when sending XML documents form SAP xMII to SAP XI. Please refer to the example XI content provided in the package or the SAP Exchange Infrastructure documentation for further details.

SSL from SAP XI to SAP xMII

In order to establish SSL between SAP XI and SAP xMII you have to configure the SAP Web Application server as SSL client with respect to the Microsoft IIS server, where SAP xMII runs on. Once you have SSL support established between those two servers you can call any SAP xMII URL via https. SAP xMII itself does not require any additional settings on its own beyond that.

Copyright