Financial Process Integrator Guide

Version 2004.5

September 2004

Contents

1 What’s New in this Release

The following changes have been introduced in the Financial Process Integrator, Version 2004.5:

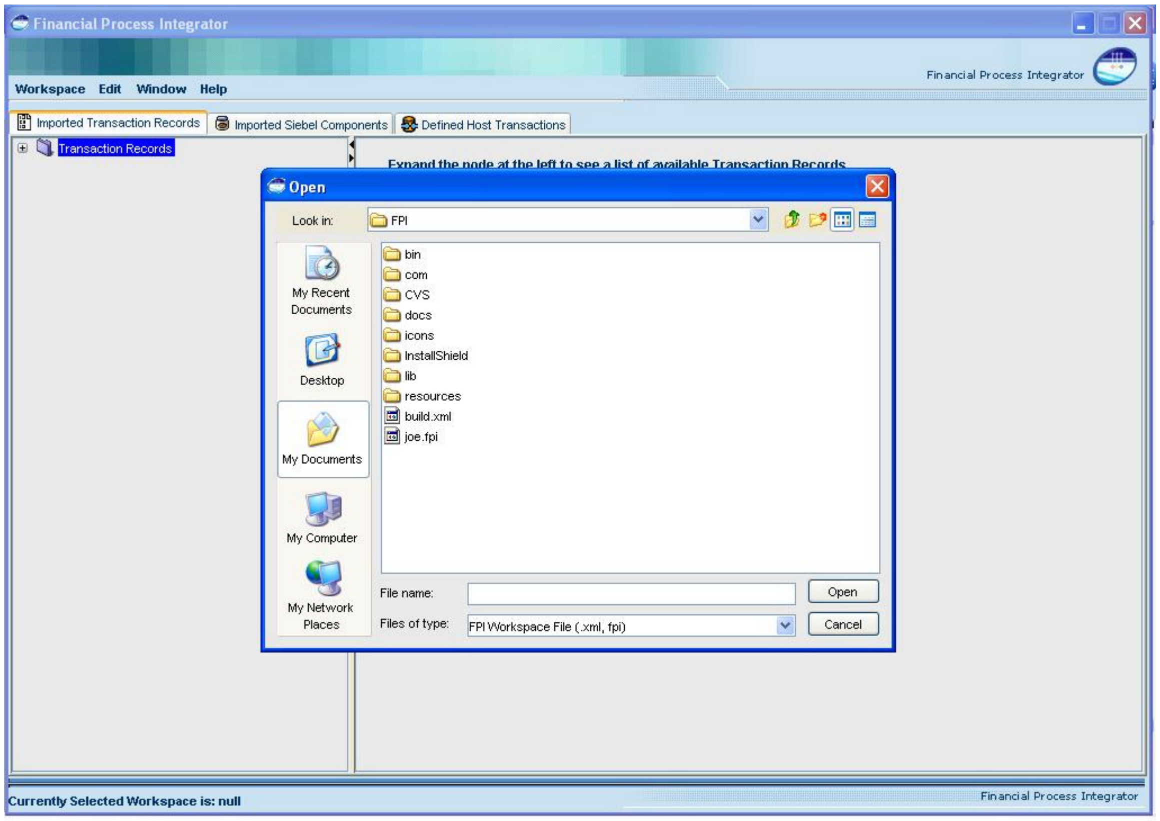
Topic	Description
Enhanced usability has been added to improve the mapping capability of the Fianancial Process Integrator tool in relation to transaction responses, pages 21 - 23	Changes have been added to facilitate the following: ability to map a host field to more than one Siebel entity attribute, option to not to map a host attribute, handling of recurring fields & groups of fields, facility for multiple occurring fields to map to either the same attribute across many instances of the same entity or different attributes of the same entity, supports nesting of multiple occurring fields within multiple occurring groups, colour coding of mapping to highlight shared entity use, facility for deleted mapping rows to be restored
Support for XML record types has been added, pages 26 – 31. See also the example which has been added, pages 41 - 48	Support for XML Transaction Records has been added. The XML record definition must adhere to the FPITransaction dtd
A Section on FPI Configuration has been added, pages 48 - 49	This section describes Financial Process Integrator XML attributes and database settings which are configured in the BankframeConstants.properties file

2 Introduction

The Financial Process Integrator tool can be used to import, define and manipulate software components that are required to map Financial Components to various Host Transaction Record formats. It allows a user to specify how these transaction mappings are defined, and then generate the SQL meta-data that is required for the MCA Services Financial Process Integrator to execute these transactions. The meta-data 'maps' a host transaction, in the form of a set of host transaction records, to a set of financial objects. Put another way, the metadata makes it possible for a client to access only enterprise java beans and have the data retrieved from a host system back-end. The scope of this document assumes that the reader is already familiar with the make-up of a standard Financial Component, and has a working knowledge of MCA Services, and in particular the MCA Financial Process Integrator engine (see the MCA Services documentation for same).

3 Getting Started

When the Financial Process Integrator is launched the following screen is displayed:


Financial Process Integrator
Workspace Edit Window Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Transaction Records
Expand the node at the left to see a list of available Transaction Records
Open
Look in: FPI
My Recent Documents
Desktop
My Documents
My Computer
My Network Places
bin
com
CVS
docs
icons
InstallShield
lib
resources
build.xml
joe.fpi
File name:
Files of type: FPI Workspace File (.xml, fpi)
Open
Cancel
Currently Selected Workspace is: null
Financial Process Integrator

This dialogue box is used to open a previously saved workspace or design model XML file. Click on the ‘Cancel’ button to continue without opening a saved workspace. Alternatively the tool tracks recently opened files. These can be accessed by clicking on the “File” menu option. At the bottom of this menu tree will appear the most recently opened workspaces.

The FPI application window is split up into three main sections. Close to the top there is a menu bar with four menus – Workspace, Edit, Window and Help. These menus will launch wizards to help a user create or manipulate objects within the application. Just below the main menu items, there is a window containing three tabs, each one with a tree-style view that displays all of the objects used by the Financial Process Integrator. The left window displays the tree view of all the components currently within your application. The right window will display a list of properties associated with the node you have selected in the tree on the left. Each node within the trees can be right-clicked to launch a pop-up menu displaying all of the actions available to the user for that particular node (these actions are equivalent to the ones listed in the menu above). There is a divider bar between the left and right windows that can slide horizontally, to allow more room on either side. It can also be fully expanded or contracted to either side by clicking on the arrow buttons on the divider.

4 Workspace

The Financial Process Integrator uses the concept of a Workspace to represent all of the data and objects that have been loaded and manipulated within the application. All of this information can be saved to a file on disk, in XML format. Therefore, any time you import files, create new objects or update mappings within the FPI tool, you can save all of the current information to a file for later use. On initial launching of the Financial Process Integrator, a default Workspace is created, which contains no data.

4.1 Workspace Operations

The Financial Process Integrator Workspace contains the following menu options

4.1.1 Creating a New Workspace File

The New menu item clears all data that is currently in the FPI's workspace. The user may be prompted to save any existing data before it is cleared.

4.1.2 Opening a Workspace File

The Open menu item allows a user to load a previously saved workspace file. All FPI workspace files are saved in XML format, with the extension .xml or .fpi.

4.1.3 Saving a Workspace File

The Save/Save AS menu items allow a user to save all data that is currently in the Host Tools workspace. This includes any imported models, imported Transaction Records, user created systems, mappings etc. Workspace files are saved in XML format on a locally accessible file system. Saved files can be loaded back into the tool at any time.

4.1.4 Generating SQL

The Generate SQL menu item allows a user to generate the SQL Insert statements based on the mappings defined in the tool for the MCA Financial Process Integrator metadata.

4.1.5 Testing Transactions

The Test Transaction menu item allows a user to test host transactions that have been defined in the Financial Process Integrator. Testing a transaction will create and send a HTTP request to a deployed instance of MCA Services and display the results in the tool.

5 Host Systems

The Host Systems list is the root node under the third tab which is titled ‘Defined Host Transactions’. The Host Systems node is a placeholder list for all Host Systems that you have defined within your workspace. Beneath the Host Systems node, there will be a list of zero or more Host Systems that have been previously added. Each one will be prefixed by the label “System:”, followed by the name given to the Host System. A Host System typically represents a physical machine that is used to host one or more applications within a financial institution. An example of a Host System could be an IBM pSeries or RS/6000. In a physical environment, these systems can host multiple sub-systems or containers for different lines of business applications.

5.1 Host System Operations

When the Host System node is selected the following menu items are available to the user:

5.1.1 Creating a New Host System

The New Host System menu item will launch a wizard that allows you to specify a new host system. To specify a host system, you will first need to select the type of transaction record format that is used by the host system. A drop-down list on the first wizard screen will provide a list of entries to choose from. The next screen in the wizard will prompt you to enter the following settings:

Host Name – A logical name for identification purposes.

Description – A description of the host system.

Host Vendor – The name of the vendor/manufacturer of the host system.

5.1.2 Generating Metadata for Host Systems, Sub Systems and Transactions

This Generate SQL menu item will generate the metadata required by the MCA Services Financial Process Integrator in SQL format. This generation is based upon the mappings that a user defines in the Request and Response nodes for a Host Transaction. The metadata will be generated for all Host Systems, SubSystems and Transactions that are defined in the workspace. The wizard will prompt you to enter the following property:

Database Vendor – Select from a list of available vendors. This ensures the SQL syntax complies with the corresponding database server being used.

When selecting any existing Host System objects in the left tree, the following actions (menu items) are available to the user:

5.1.3 Adding a Sub System

The Add SubSystem menu item will launch a wizard that allows you to specify a new host subsystem. To specify a host subsystem, you will be prompted to enter the following settings:

Host System – Select an existing Host System that your Sub System will be associated with.

SubSystem Name– A logical name of the Sub System for identification purposes.

Description – A description of the host sub system.

5.1.4 Removing a Host System

The Remove menu item will remove the host system that is currently selected from the workspace. When a host system is removed, it will also remove all child node objects that are below it, therefore any Host Sub Systems, Host Transactions and Mappings that have been added to this Host System will also be removed. The user will be prompted with a warning, asking them if they wish to remove all child nodes in this process.

5.1.5 Importing Host Connectors

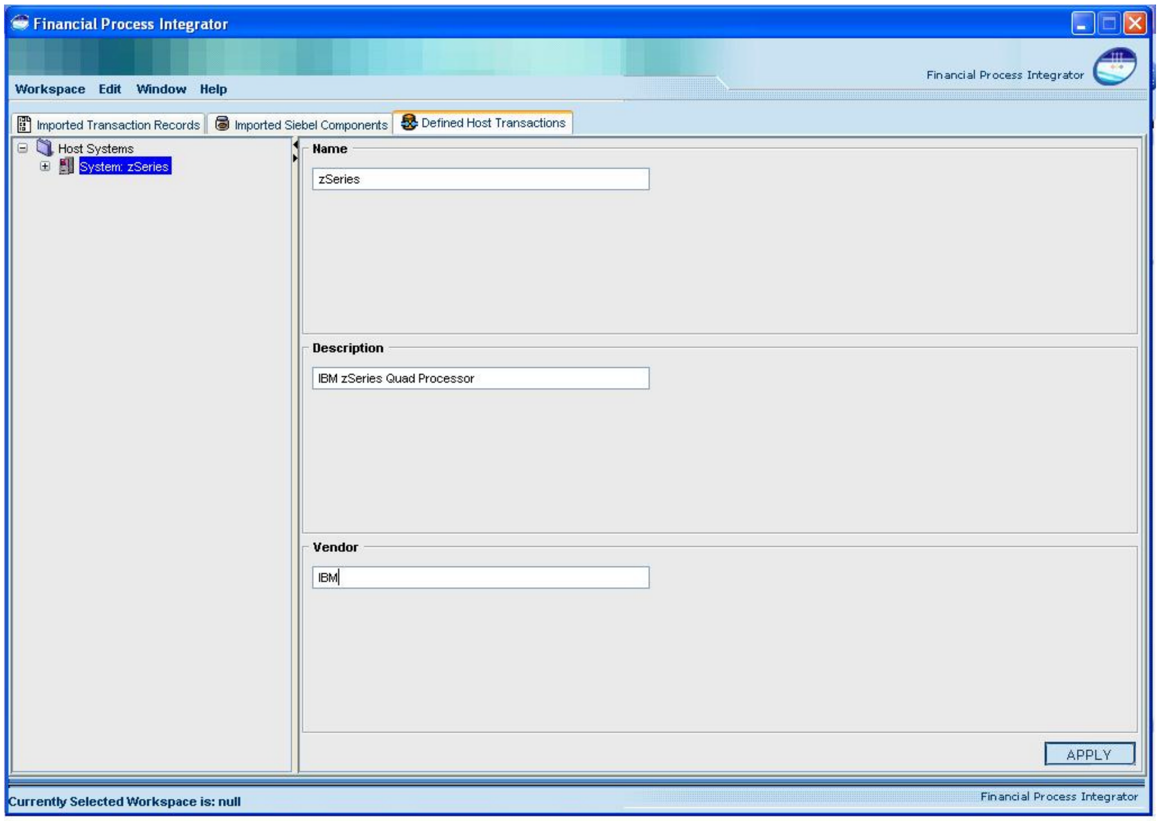
The Import Connectors menu item will launch a wizard that allows you to import Host Connectors into the FPI workspace. Host Connectors are defined in the main configuration file used by MCA Services. This file is named `BankframeResource.properties`, and is usually located in the directory of the application where an instance of MCA Services has been deployed. It might also be available from the repository.

5.2 Host System Properties

The Host System list has no properties, but each individual host system will have properties associated with it, which can be viewed in the right window of the Integration tool when one is selected. These properties would have been specified when the Add Host System wizard was launched. Properties that are editable can be changed at any time by modifying the corresponding text field in the right window. You must click the Apply button in the bottom right corner to apply any changes you have made. Each Host System node has the following properties associated with it:

Name	A logical name for identification purposes.
Description	A description of the host system.
Vendor	The name of the vendor/manufacturer of the host system.

The following screen shot shows the property window when a Host System instance is selected:


Financial Process Integrator
Workspace Edit Window Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Host Systems
System: zSeries
APPLY
Currently Selected Workspace is: null

Name: zSeries

Description: IBM zSeries Quad Processor

Vendor: IBM

6 Sub Systems

A new node will be added beneath a Host System node when a user completes the add sub system wizard. Each node will be prefixed by the label 'Sub-System', followed by the name given to the Host Sub System. A Host Sub System typically represents a software server instance, such as a container or database that is used to host a software application within a bank. An example of a Host Sub System could be a Branch Sub System, or an Internet Banking Sub System. A sub system is often tied to a particular line of business within the bank environment.

6.1 Sub System Operatins

When the Host Sub System node is selected, the following menu items are available to the user:

6.1.1 Creating a Host Transaction

The New Host Transaction menu item will launch a wizard that allows you to create a new host transaction. To specify a host transaction, you will be prompted to enter the following settings:

Txn Type – A user defined transaction type to help associate this transaction with a middleware or legacy system. The host connector name is often used here.

Description - A description of the host system.

Txn Name – The name of the transaction.

Host System – Select a Host System from the list provided.

Sub System – Select a Host Sub System from the list provided

Host Connector – Select a Host Connector from the list provided. If the list is empty, you should cancel the wizard and import a `BankFrameResource.properties` file to add connectors to your workspace first. Or, complete the wizard with no connector specified, then change the Host Transaction properties later by selecting the connector and clicking the Apply button.

Request Record – Select a Transaction Record from the list provided. This record will be used as the request data that is sent to the host. If the list is empty, you will have to cancel the wizard and import some Transaction Records.

Response Record – Select a Transaction Record from the list provided. This record will be used as the response data that is returned from the host. If the list is empty, you will have to cancel the wizard and import some Transaction Records.

6.1.2 Removing a Host Sub System

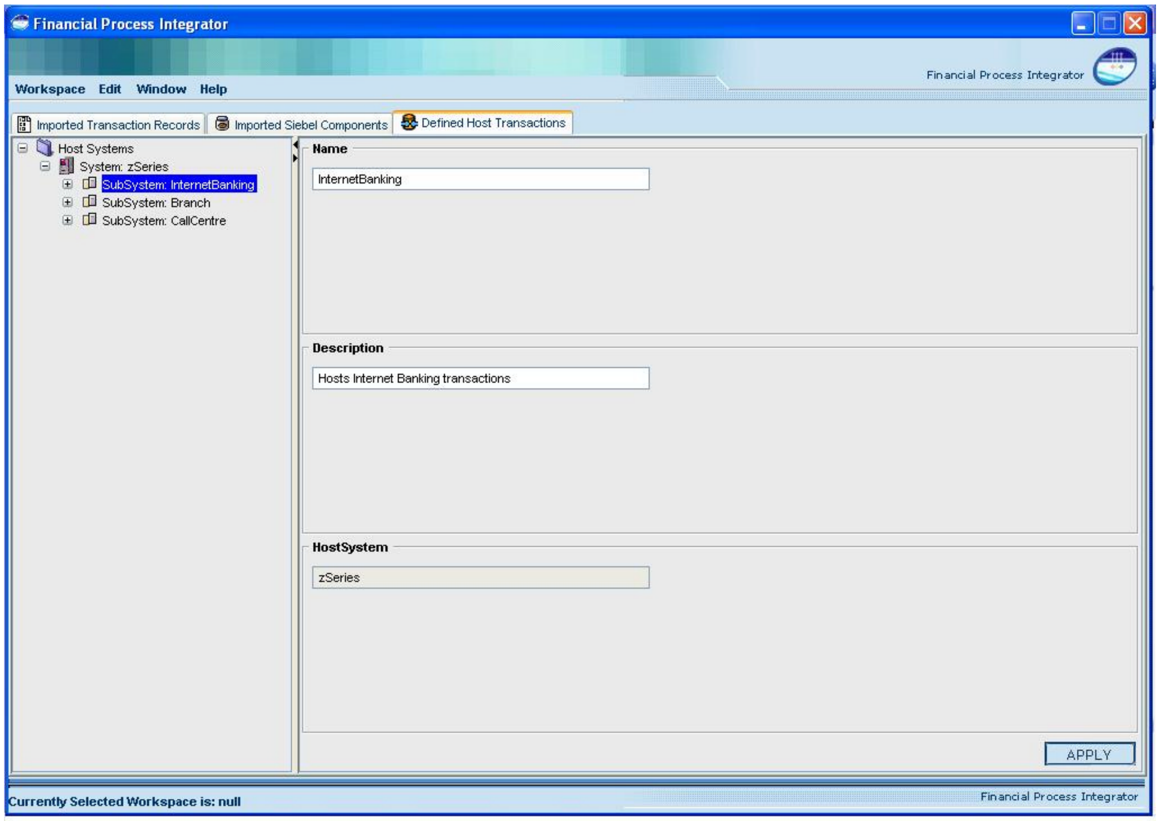
The Remove menu item will remove the host sub system that is currently selected from the workspace. When a host sub system is removed, it will also remove all child node objects that are below it, therefore any Host Transactions and Mappings that have been added to this Host Sub System will also be removed. The user will be prompted with a warning, asking them if they wish to remove all child nodes in this process.

6.2 Sub System Properties

The Host Sub System node has properties associated with it, which can be viewed in the right window of the Host Tool when a Host Sub System node is selected. These properties were specified when the Add Host Sub System wizard was launched. Properties that are editable can be changed at any time by modifying the corresponding text field in the right window. You must click the Apply button in the bottom right corner to apply any changes you have made. Each Host Sub System node has the following properties associated with it:

Name	A logical name for identification purposes.
Description	A description of the host sub system.
Host System	The name of the host system that this sub system is associated with.

The following screen shot shows the properties window when a Host Sub System is selected in the tree:


Financial Process Integrator
Financial Process Integrator
Workspace
Edit
Window
Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Host Systems
System: zSeries
SubSystem: InternetBanking
SubSystem: Branch
SubSystem: CallCentre
APPLY
Financial Process Integrator

Name
InternetBanking

Description
Hosts Internet Banking transactions

HostSystem
zSeries

Currently Selected Workspace is: null

7 Host Transactions

A Host Transaction node will be added beneath a Sub System node when a user adds a new transaction and completes the add transaction wizard. Each transaction node will be prefixed by the label 'Transaction:', followed by the name of the Host Transaction. A Host Transaction represents a single transaction that will be made to the host system, from your Java application. The transaction is made up of a request record and a response record, selected from the list of current transaction records that have been imported into the workspace. After selecting the records used in the transaction, you will then specify how your business objects will be mapped to these transaction records. An example of a Host Transaction could be a Bill Payment transaction. Once you have defined and mapped your host transaction, you can then use the Financial Process Integrator to generate the SQL statements required to populate the metadata used by the MCA Financial Process Integrator.

7.1 Host Transaction Operations

When a Host Transaction node is selected, the following menu items are available to the user:

7.1.1 Removing a Host Transaction

The Remove menu item will remove the host transaction that is currently selected from the workspace. When a host transaction is removed, it will also remove all mappings associated with the transaction. The user will be prompted with a warning, asking them if they wish to remove all child nodes in this process.

7.1.2 Defining a Persister Mapping for a Host Transaction

The Define Persister menu item will launch a wizard that lets you define a Persister mapping for a Host Transaction. A Persister class is used to persist Financial Components to a Host System. More information on Persisters can be found in the MCA Services Developers Guide.

7.1.3 Testing a Host Transaction

The Test Transaction menu item will allow a user to test host transactions that have been defined in the Financial Process Integrator. Testing a transaction will create and send a HTTP request to a deployed instance of MCA Services and display the results.

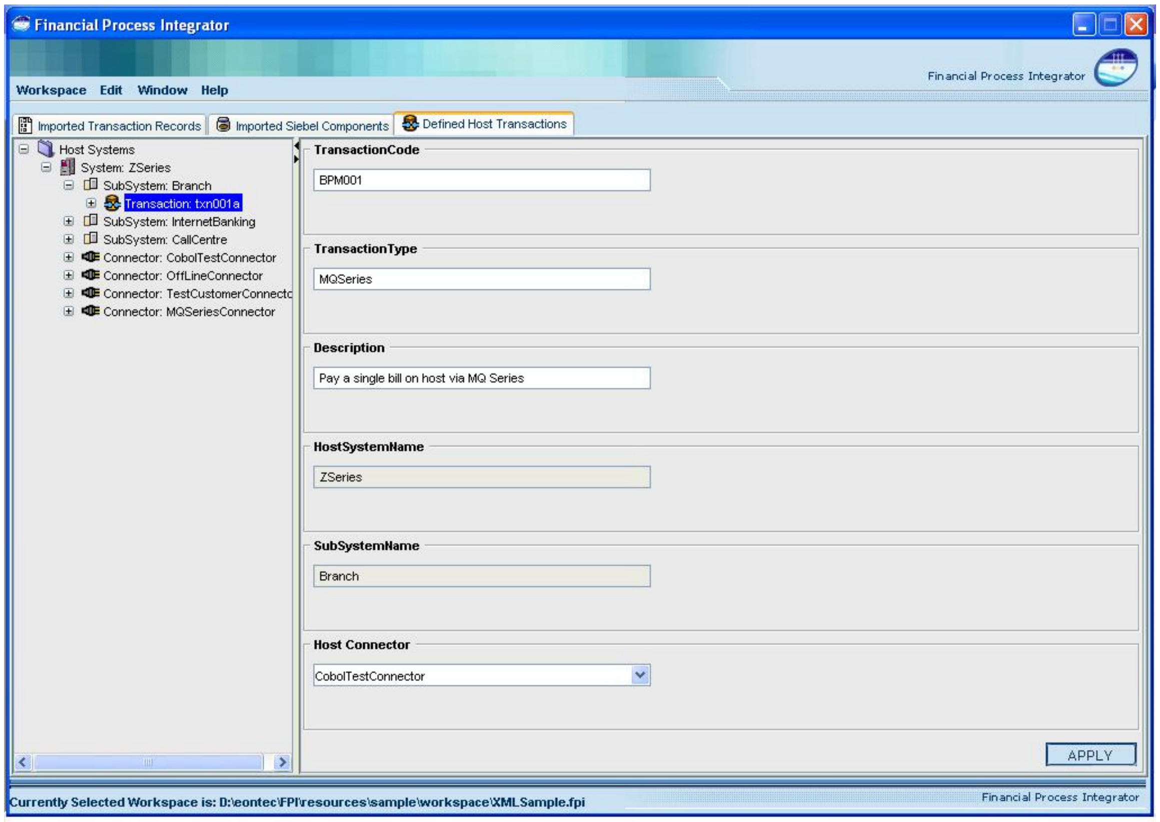
7.2 Host Transaction Properties

The Host Transaction node has properties associated with it, which can be viewed in the right window of the Host Tool when one is selected. These properties were specified when the Add Host Transaction wizard was launched. Properties that are editable can be changed at any time by modifying the corresponding text field in the right window. You must click the Apply button in the bottom right corner to apply any changes you have made. Each Host Transaction node has the following properties associated with it:

TransactionCode	User defined code for this transaction.
TransactionType	User defined type for this transaction.

Description	A description of the host transaction.
Host System Name	Host System associated with the Transaction.
Sub System Name	Host Sub System associated with the Transaction.
Host Connector	Host Connector used by this Transaction.

The following screen shot shows the properties window when a host transaction is selected in the tree:


Financial Process Integrator
Workspace Edit Window Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Host Systems
System: ZSeries
SubSystem: Branch
Transaction: txn001a
SubSystem: InternetBanking
SubSystem: CallCentre
Connector: CobolTestConnector
Connector: OffLineConnector
Connector: TestCustomerConnectc
Connector: MQSeriesConnector
APPLY

TransactionCode: BPM001

TransactionType: MQSeries

Description: Pay a single bill on host via MQ Series

HostSystemName: ZSeries

SubSystemName: Branch

Host Connector: CobolTestConnector

Currently Selected Workspace is: D:\eontec\FPI\resources\sample\workspace\XMLSample.fpi

8 Persisters

When entity beans are used to model data on host system they must be implemented using Bean Managed Persistence (BMP). To do this they must interact with the MCA Financial Process Integrator services. The task of communicating with the MCA Financial Process Integrator is delegated to a helper object. This helper object is called a 'Persister' object. A new Persister node will be added beneath a Host Transaction node when a user completes the add Persister wizard. Each one will be prefixed by the label ‘Persister:’ followed by the method name of the Financial Component that is used to talk to the Persister. The properties defined in your Persister will be used to generate the `PERSISTER_TXN_MAP` table of the metadata. When defining a Persister you must select a Financial Object (Entity Bean) as well as the method on that object that is being called. You must also select a cache policy, which determines whether the data from the MCA Financial Process Integrator is cached or not. The cache policy should be set to `none` if the transaction results cannot be cached, to `persistent` if the cache is to be written to a database so it is available even if there is a system failure or to `memory` if it is to be cached in memory. The `TIME_OUT_VALUE` attribute that you must enter in the wizard specifies the length of time (in milliseconds) that the stored data remains valid. When data is retrieved from the cache its creation time is compared to the current time and if the difference is greater than the `TIME_OUT_VALUE` then data is requested from the host.

8.1 Persister Operation

When a Persister node is selected the following action is available to the user:

8.1.1 Removing a Persister

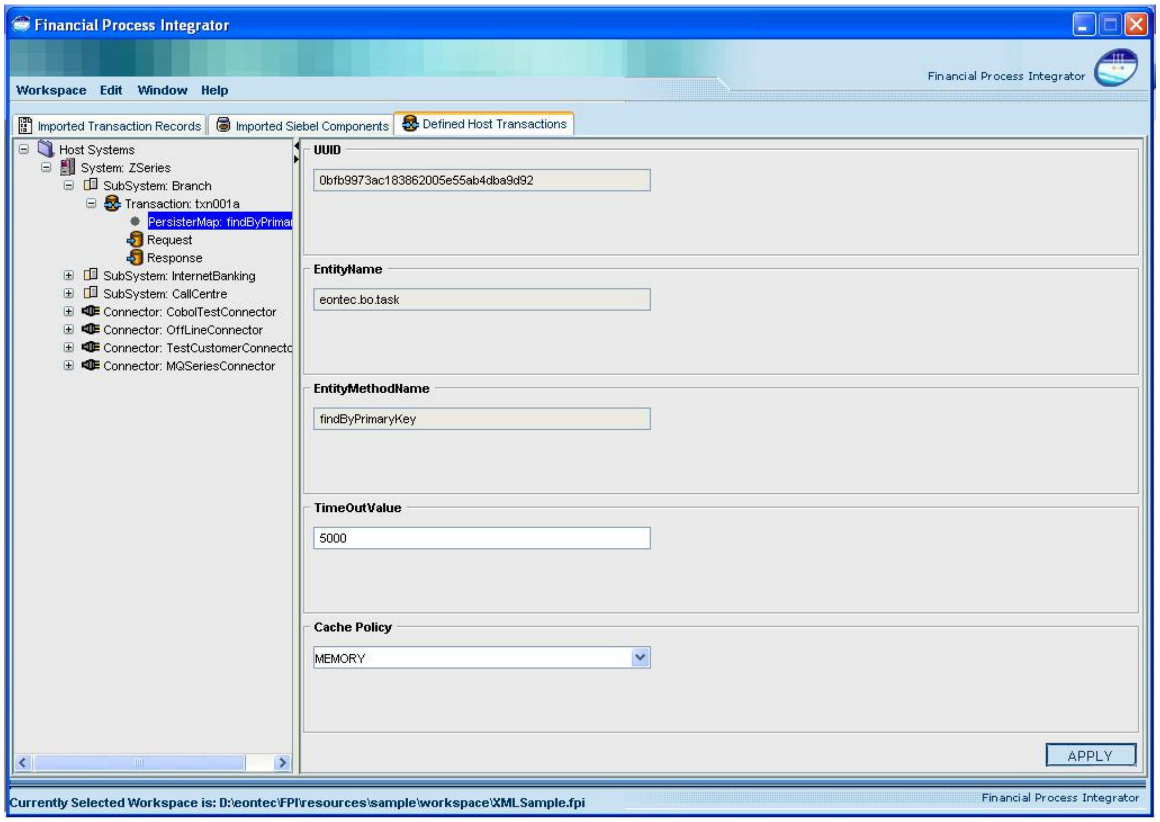
The Remove menu item will remove the Persister that is currently selected from the workspace.

8.2 Persister Properties

Each Persister node has properties associated with it, which can be viewed in the right window of the Host Tool when one is selected. These properties were specified when the Define Persister wizard was launched. Properties that are editable can be changed at any time by modifying the corresponding text field in the right window. You must click the Apply button in the bottom right corner to apply any changes you have made. Each Persister node has the following properties associated with it:

Entity Name	Name of the Financial Object (Entity) for this Persister
Entity Method Name	Method name of the entity.
Time Out Value	Time out value (in milliseconds) for the cache policy.
Cache Policy	Type of caching policy to be used.

The following screen shot shows the properties window when a Persister is selected in the tree:


Financial Process Integrator
Workspace Edit Window Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
APPLY

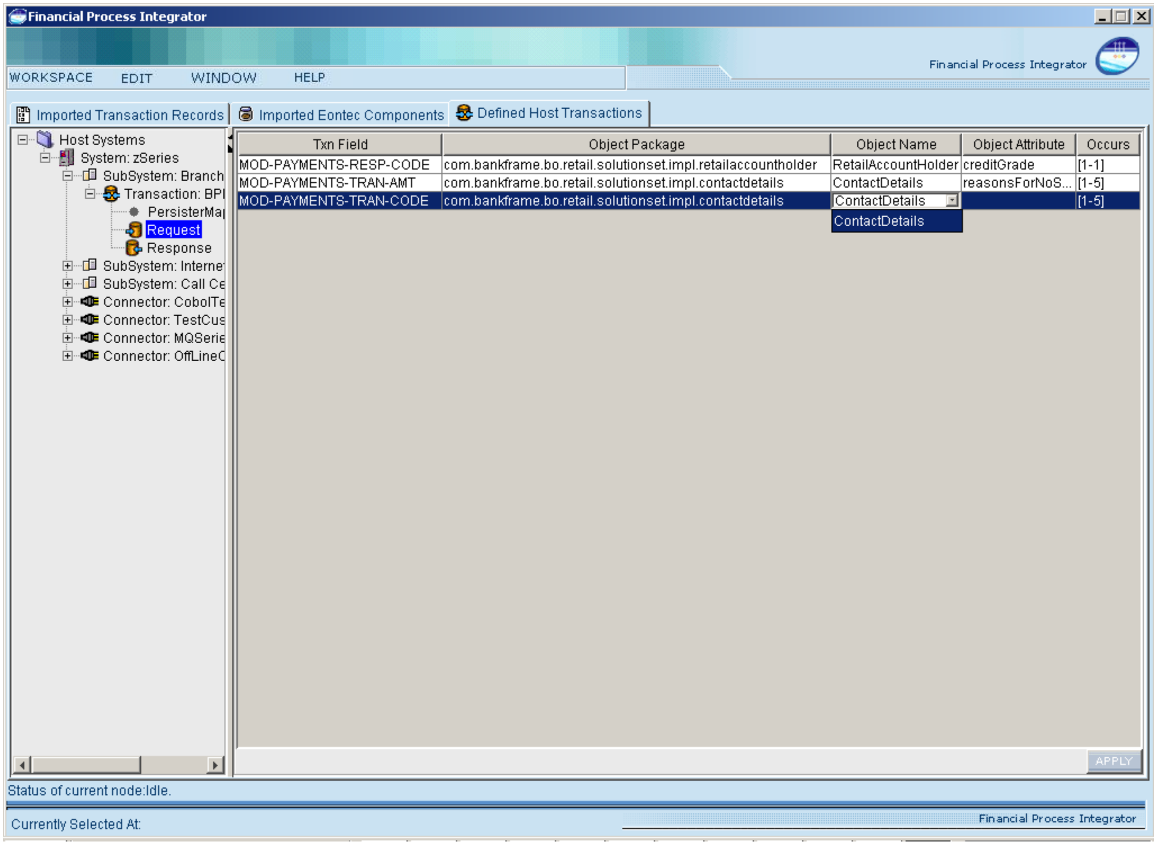
```
Host Systems
  System: ZSeries
    SubSystem: Branch
      Transaction: txn001a
        PersisterMap: findByPrimar
        Request
        Response
    SubSystem: InternetBanking
    SubSystem: CallCentre
    Connector: CobolTestConnector
    Connector: OffLineConnector
    Connector: TestCustomerConnectc
    Connector: MQSeriesConnector
```

UUID	0bfb9973ac183862005e55ab4dba9d92
EntityName	eontec.bo.task
EntityMethodName	findByPrimaryKey
TimeOutValue	5000
Cache Policy	MEMORY

Currently Selected Workspace is: D:\eontec\FPI\resources\sample\workspace\XMLSample.fpi

9 Transaction Request

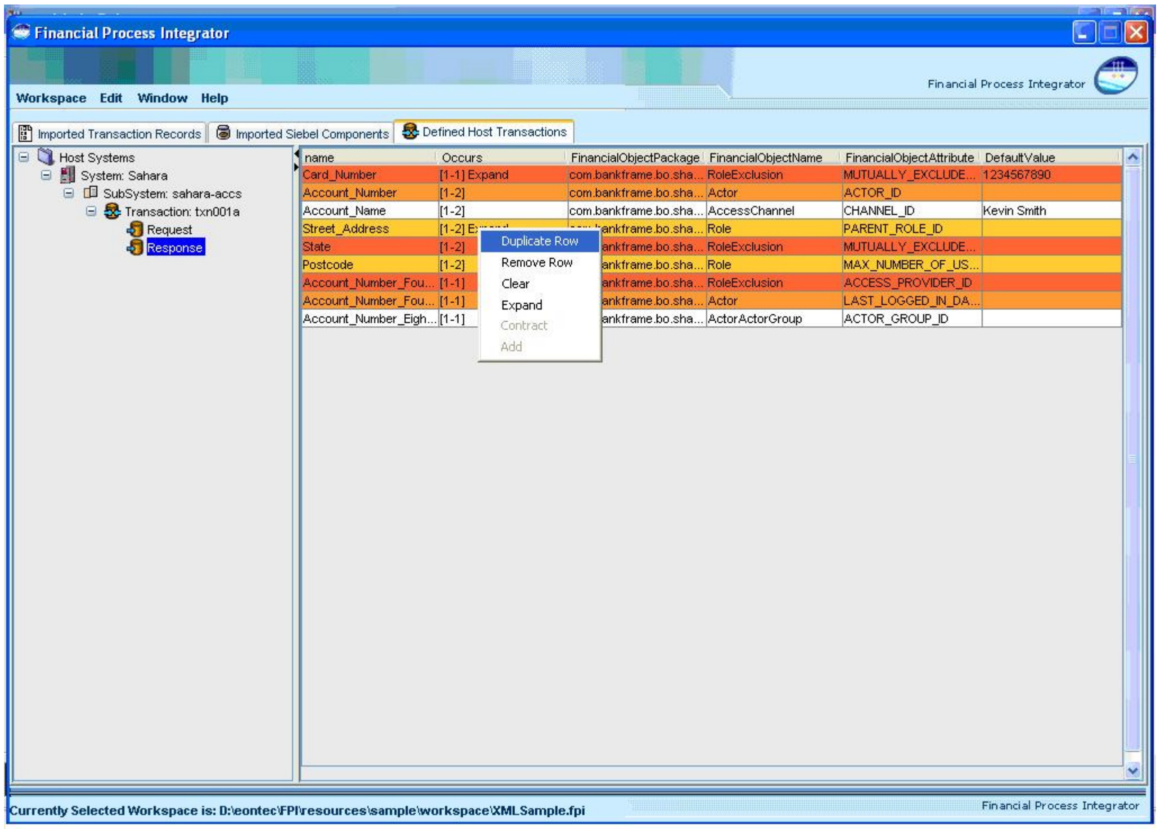
Directly beneath the Host Transaction node are Request and Response nodes that are automatically added when the Host Transaction Wizard is completed. When selecting the Request node in the left tree, a mapping table will be drawn in the right window of the application. This table will be used to map fields in your Request transaction record to attributes of your business objects. When the table is refreshed, you will notice that it contains rows corresponding to fields in the transaction record you selected for the host request. For each field (row) in this table, you must fill in the corresponding column values to reflect how you wish to map your transactions. This will involve selecting java packages, financial objects and attributes that map to the transaction record field. For Cobol Copybook transactions and the xml transaction records, there is an `occurs` column in this table that indicates whether or not a field in the original transaction record occurs (repeats) multiple times.The divider bar between the left and right windows can be moved to provide a bigger view of your mapping table. It is also 'one-touch expandable', which means that clicking on the arrows shown on the divider bar will expand one window to the full application size, and temporarily hide the other window.
The screen shot below shows part of the application window when the Request node is selected in the tree, and the mapping table is created in the right window. Note that the request object shown below only has two transaction fields to map, and the mapping window has been resized to better see all of the columns in the table.



Txn Field	Object Package	Object Name	Object Attribute	Occurs
MOD-PAYMENTS-RESP-CODE	com.bankframe.bo.retail.solutionset.impl.retailaccountholder	RetailAccountHolder	creditGrade	[1-1]
MOD-PAYMENTS-TRAN-AMT	com.bankframe.bo.retail.solutionset.impl.contactdetails	ContactDetails	reasonsForNoS...	[1-5]
MOD-PAYMENTS-TRAN-CODE	com.bankframe.bo.retail.solutionset.impl.contactdetails	ContactDetails		[1-5]

10 Transaction Response

The mapping table of the Response node is slightly different than the request node. When selecting the Response node in the left tree, a mapping table will be drawn in the right window of the application.
This table will be used to map fields in your Response transaction record to attributes of your business objects.
For each field (row) in this table, the user selects a package, financial object and financial object attribute that maps to the transaction record field. The table cells for these properties are drop-down lists that are populated from all of the business objects that are currently stored in the application server repository. The `occurs` column in this table indicates whether or not a field in the original transaction record occurs (repeats) multiple times.
The following screen shot shows part of the application window when a Response node is selected:


Financial Process Integrator
Workspace
Edit
Window
Help
Financial Process Integrator
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Host Systems
System: Sahara
SubSystem: sahara-accs
Transaction: txn001a
Request
Response
Duplicate Row
Remove Row
Clear
Expand
Contract
Add
Financial Process Integrator

name	Occurs	FinancialObjectPackage	FinancialObjectName	FinancialObjectAttribute	DefaultValue
Card_Number	[1-1] Expand	com.bankframe.bo.sha...	RoleExclusion	MUTUALLY_EXCLUDE...	1234567890
Account_Number	[1-2]	com.bankframe.bo.sha...	Actor	ACTOR_ID	
Account_Name	[1-2]	com.bankframe.bo.sha...	AccessChannel	CHANNEL_ID	Kevin Smith
Street_Address	[1-2] E	ankframe.bo.sha...	Role	PARENT_ROLE_ID	
State	[1-2]	ankframe.bo.sha...	RoleExclusion	MUTUALLY_EXCLUDE...	
Postcode	[1-2]	ankframe.bo.sha...	Role	MAX_NUMBER_OF_US...	
Account_Number_Fou...	[1-1]	ankframe.bo.sha...	RoleExclusion	ACCESS_PROVIDER_ID	
Account_Number_Fou...	[1-1]	ankframe.bo.sha...	Actor	LAST_LOGGED_IN_DA...	
Account_Number_Eigh...	[1-1]	ankframe.bo.sha...	ActorActorGroup	ACTOR_GROUP_ID	

Currently Selected Workspace is: D:\eontec\FPI\resources\sample\workspace\XMLSample.fpi

If a response transaction field is to be mapped to more than one Business object attribute the user highlights the response transaction field, right clicks on this row and selects “Duplicate Row” from the drop down menu. The user can now map this transaction field to another Business object attribute. This allows a transaction field to initialize the state of many diiferent Business object attributes.
Alternatively, if the user is not interested in a particular response transaction field being returned from the host transaction, the user highlights the response transaction field and selects “Remove Row”.

10.1 Transaction Response Operations

By right clicking on the response mapping screen the following menu options are available:

10.1.1 Duplicating a Response Transaction Field

The Duplicate Row menu item enables the user to duplicate the currently selected response transaction field

10.1.2 Removing a Response Transaction Field

The Remove Row menu item enables the user to remove the currently selected response transaction field

10.1.3 Removing Response Transaction Field Mapping

The Clear menu item enables the user to remove the mapping information for the currently selected response transaction field if a mapping exists for the field

10.1.4 Displaying All Occurrences of a Transaction Field

The Expand menu item enables the user to expand the currently selected response transaction field to display all occurrences of the field

10.1.5 Displaying One Instance of a Transaction Field

The Contract menu item enables the user to contract the currently selected response transaction field to display one single occurrence of the field

10.1.6 Restoring a Deleted Response Transaction Field

The Add menu item enables the user to, where applicable, display and restore the response transaction fields that where previously removed

10.2 Handling Recurring Fields

The above drop down menu list available in the response mapping screen presents the user with 6 options. 3 of these options (Expand, Contract, Add) are related to the handling of recurring fields. For instance, if a transaction response field is defined such that it has a field occurrence value of n, the field can be mapped to a particular business object attribute and the mapping can be configured to ensure that each occurrence of the field maps to a new instance of the same business object attribute. This is achieved by having the field marked as “Expand” on the response mapping screen. However, if you would like each occurrence of the recurring field to map to a different business object attribute select “Expand” on the recurring field and map each occurrence as required. Select “Contract” on the said field to return to previous state, i.e. each occurrence of the field maps to a new instance of the same Business object attribute.

Note: Recurring fields are also supported within groups, i.e. recurring fields within groups or recurring groups.

Finally, if the user has removed a response transaction field that they require to be added back in to the mapping screen they can select “Add” from the drop down list and select the field that they previously removed.

The divider bar between the left and right window panes can be moved to provide a larger view of your mapping table. It is also ‘one-touch expandable’, which means that clicking on the arrows shown on the divider bar will expand one window to the full application size, and temporarily hide the other window.

11 Host Connectors

The Host Connectors in the workspace are added beneath the Host System node after a user has imported a `BankFrameResource.properties` file. A Host Connector represents a software application or server that is used as a mediator for communicating to a Host System. The MCA Financial Process Integrator specifies a connector that is used to transform and forward messages from your application through the middleware and on to the Host System. Host Connector information is specified in the `BankFrameResource.properties` file, and therefore most of the connector attributes are not editable, with the exception of the DataFormatter class, and a list of key-value pairs in a string format.

11.1 Host Connector Operations

The following actions are available when the Connector node is selected:

11.1.1 Removing a Host Connector

Selecting the Remove menu item will remove the selected Host Connector from the workspace

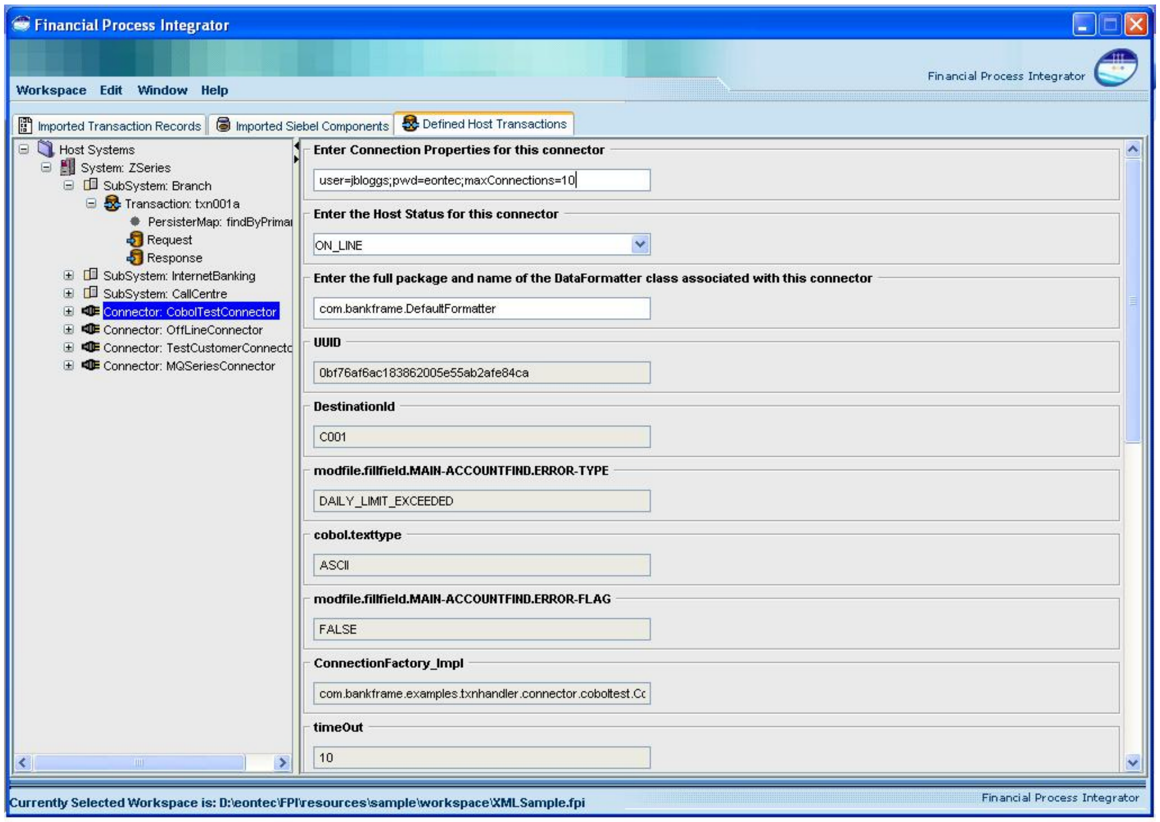
11.2 Host Connector Properties

Each connector will have attributes associated with it, which can be viewed in the right window of the Host Tool when one is selected. These properties were parsed when the Import Connectors wizard was launched. Imported properties are not editable, but you can define a Properties list (key/value pairs separated by semi-colons), and can also specify a DataFormatter class associated with a connector. These two values can be changed at any time by modifying the corresponding text field in the right window. You must click the Apply button in the bottom right corner to apply any changes you have made. Each Host Connector node has the following standard properties associated with it:

Properties	A list of key/value property pairs that are required by the connector for this middleware service. Entries should be of the form **key=value;**
DataFormatter	The full package and class name of a DataFormatter class that is associated with this connector.

In addition to the properties listed above, each connector will have a list of connector specific properties that are not editable.

The following screen shot shows the properties window when a connector node is selected:


Financial Process Integrator
Workspace Edit Window Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Host Systems
System: ZSeries
SubSystem: Branch
Transaction: txn001a
PersisterMap: findByPrimai
Request
Response
SubSystem: InternetBanking
SubSystem: CallCentre
Connector: CobolTestConnector
Connector: OffLineConnector
Connector: TestCustomerConnectc
Connector: MQSeriesConnector

Enter Connection Properties for this connector

user=jbloggs;pwd=eontec;maxConnections=10

Enter the Host Status for this connector

ON_LINE

Enter the full package and name of the DataFormatter class associated with this connector

com.bankframe.DefaultFormatter

UUID

0bf76af6ac183862005e55ab2afe84ca

DestinationId

C001

modfile.fillfield.MAIN-ACCOUNTFIND.ERROR-TYPE

DAILY_LIMIT_EXCEEDED

cobol.texttype

ASCII

modfile.fillfield.MAIN-ACCOUNTFIND.ERROR-FLAG

FALSE

ConnectionFactory_Impl

com.bankframe.examples.txnhandler.connector.coboltest.Cc

timeOut

10

Currently Selected Workspace is: D:\eontec\FPI\resources\sample\workspace\XMLSample.fpi

12 Transaction Records

The Transaction Records list can be found by clicking on the first tab in the application window, titled Imported Transaction Records. The Transaction Records node is a placeholder list for all Transaction Records that a user has imported from the repository. Beneath the Transaction Records node, there will be a list of zero or more Transaction Records that have been previously imported. The records are denoted by a label in front of the name that identifies the type of record it represents (e.g. “CopyBook: or XML”). A Transaction Record represents a particular record or data type that is being used by your host system. Transaction Records can be expressed via the XML transaction record definition. Alternatively, you can still import cobol copybooks to allow for backward compatability. The preferred approach is to convert the middleware type (i.e. web-methods, cobol copybooks records) into the XML record definition. The XML record defintion must adhere to the following dtd.

```
<!--FPITransaction.dtd-->
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % boolean "(True | False)">
<!ELEMENT Transaction (TransactionOverview?, (Group | Field)*)>
<!ATTLIST Transaction
        hostMiddleWare (cobol | soap | mqseries | webmethods) #REQUIRED
>
<!ELEMENT TransactionOverview (#PCDATA)>
<!ELEMENT Group (GroupOverview?, (Group | Field)*)>
<!ATTLIST Group
        groupName CDATA #REQUIRED
        groupOccurrences CDATA #REQUIRED
        redefines CDATA #IMPLIED
>
<!ELEMENT GroupOverview (#PCDATA)>
<!ELEMENT Field (FieldOverview?)>
<!ATTLIST Field
        fieldOccurrences CDATA #REQUIRED
        fieldName CDATA #REQUIRED
        fieldType CDATA #REQUIRED
        length CDATA #IMPLIED
        javaClass CDATA #IMPLIED
        javaMethod CDATA #IMPLIED
        fieldEncoding CDATA #IMPLIED
        fieldPadding CDATA #IMPLIED
        decBefore CDATA #IMPLIED
        decAfter CDATA #IMPLIED
        fieldSigned %boolean; #IMPLIED
        fieldAligned (LEFT | RIGHT) #IMPLIED
        mandatory (Yes | No) #IMPLIED
>
<!ELEMENT FieldOverview (#PCDATA)>
```

This dtd represents the cumulative field list for middleware types supported. From the BankFrameConstants.properties file (found in the resources folder) the user can see which fields are deemed as mandatory for a particular middleware type, e.g.

```
COMMON_FIELD_ATTRIBUTES=String[fieldName=REQUIRED,fieldType=REQUIRED]

COBOL_FIELD_ATTRIBUTES=String[fieldOccurrences=REQUIRED,length=REQUIRED,fieldEncoding=IMPLIED,fieldPadding=IMPLIED,decBefore=IMPLIED,decAfter=IMPLIED,fieldSigned=IMPLIED,fieldAligned=IMPLIED,mandatory=IMPLIED]
```

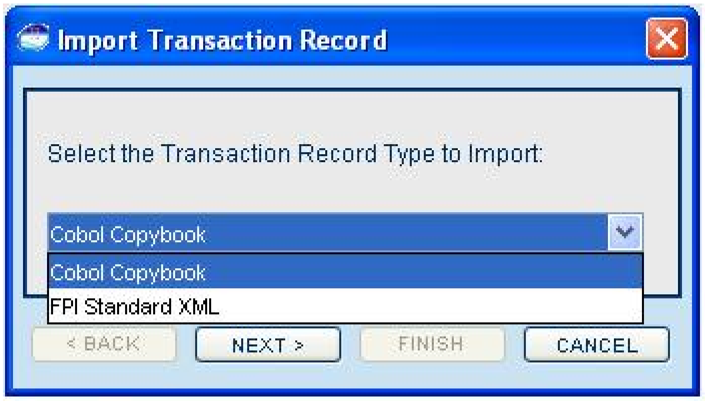
COMMON_FIELD_ATTRIBUTES represents the fields that are mandatory across all middleware types.

The fields that appear in the **COBOL_FIELD_ATTRIBUTES** that are marked as REQUIRED represent the fields that are mandatory for cobol host transactions. If a field is marked as REQUIRED and is not present in the XML transaction record (middleware type: cobol) been imported a Validation Exception will occur.

Refer to the appendix on FPI Configuration for a description of Financial Process Integrator Settings that are configured in the BankFrameConstants.properties file

12.1 Importing a Transaction Record

The Import Transaction Record menu item will launch a wizard that allows you to import a Transaction Record. Cobol Copybooks and XML Transaction records are imported in text format (either `.txt` file or `.doc` file format).


Import Transaction Record
Select the Transaction Record Type to Import:
Cobol Copybook
Cobol Copybook
FPI Standard XML
< BACK
NEXT >
FINISH
CANCEL

The FPI automatically parses the record transaction type and stores its associated properties and structure in the workspace. Individual groups and fields of the copybook are displayed as new nodes, listed below the transaction record node. Each field has many attributes associated with it, which were automatically parsed from the copybook when it gets imported. Some additional properties are added, which can be modified by the user (such as error conditions).

Remove All Records: This menu item will remove all Transaction Records that have been currently imported into the workspace. If any of the records are currently being used in a Host Transaction definition and mapping, you should go back and remove the defined transactions that contain these records, as they will be invalid.

Remove (When Individual Record is selected): This will remove the currently selected transaction record from the workspace. If the record is currently being used in a Host Transaction definition and mapping, should go back and remove the defined transaction that contains that record, as it will be invalid.

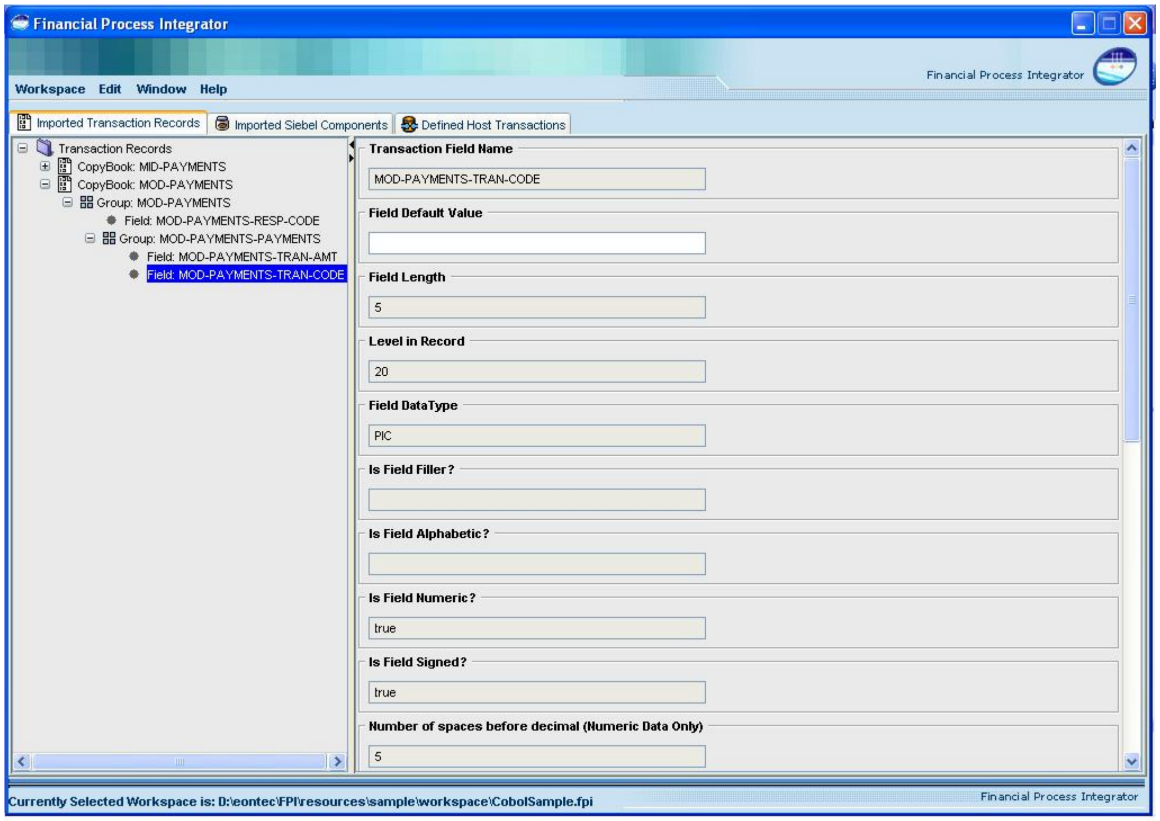
12.2 Transaction Record Properties

The Transaction Record list has no properties, but each Transaction Record as well as its sub groups, and fields will have properties associated with it, which can be viewed in the right window of the Host Tool when one is selected. These properties were parsed from the copybook or xml Transaction record document via the Import Transaction Record wizard. Properties that are editable can be changed at any time by modifying the corresponding text field in the right window. You must click the Apply button in the bottom right corner to apply any changes you have made. Note that properties will vary, depending on the make-up of the transaction record you have imported. Also, for xml Transaction records the properties displayed are related to the underlying type of the XML transaction record, i.e. the properties to be displayed for a WebMethod are specified in the BankFrameConstants.properties file (as described in the previous section). The following table illustrates the transaction fields that are applicable for the different middleware types:

Attributes	***COBOL***	***Java Wrapper / Web Methods***	***SOAP***	***MQ Series***
fieldName	Y	Y	Y	Y
fieldType	Y	Y	Y	Y
length	Y			Y
fieldEncoding	Y			Y
fieldPadding	Y			Y
decBefore	Y			Y
decAfter	Y			Y
fieldSigned	Y			Y
fieldAligned	Y			Y
javaClass		Y		
javaMethod		Y		
fieldOccurrences	Y			Y
mandatory	Y			Y

Cobol Copybooks

The following screen shot shows part of the application window when a Transaction Record attribute is selected in the tree:

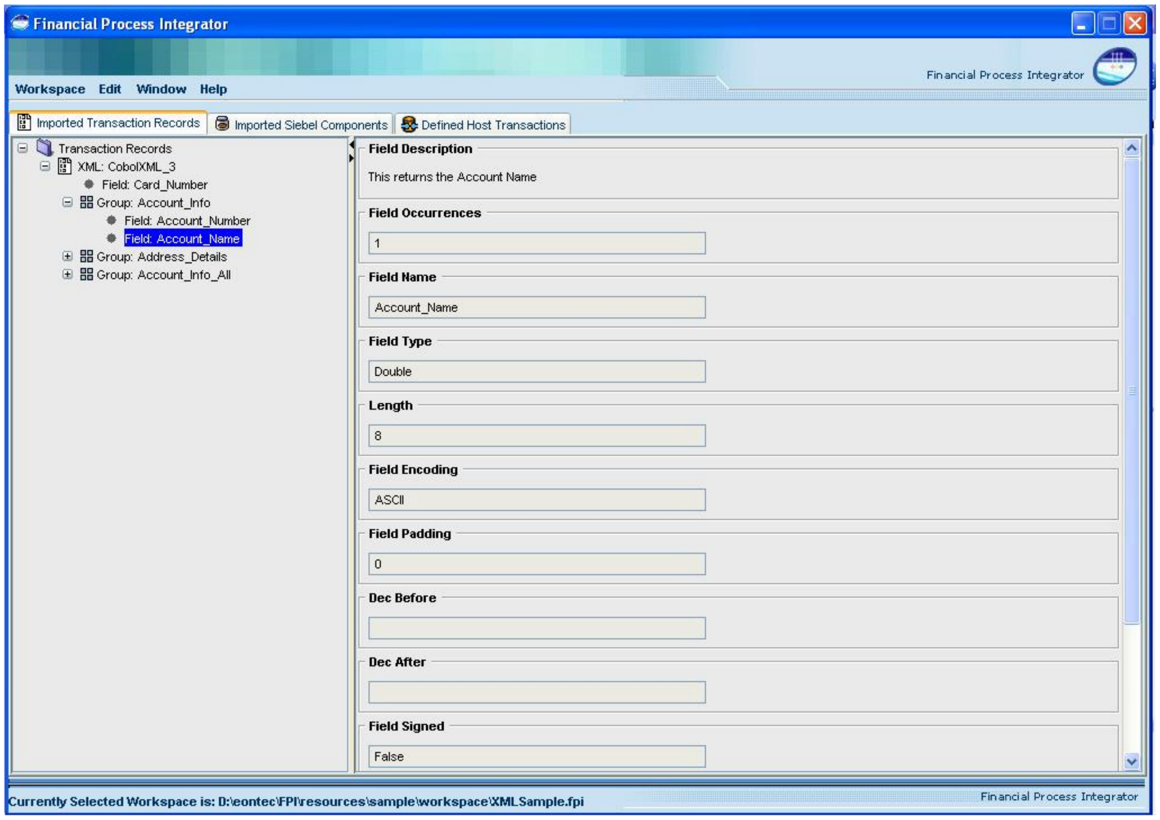

Financial Process Integrator
Workspace Edit Window Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Transaction Records
CopyBook: MID-PAYMENTS
CopyBook: MOD-PAYMENTS
Group: MOD-PAYMENTS
Field: MOD-PAYMENTS-RESP-CODE
Group: MOD-PAYMENTS-PAYMENTS
Field: MOD-PAYMENTS-TRAN-AMT
Field: MOD-PAYMENTS-TRAN-CODE

Transaction Field Name: MOD-PAYMENTS-TRAN-CODE
Field Default Value:
Field Length: 5
Level in Record: 20
Field DataType: PIC
Is Field Filler?:
Is Field Alphabetic?:
Is Field Numeric?: true
Is Field Signed?: true
Number of spaces before decimal (Numeric Data Only): 5

Currently Selected Workspace is: D:\eontec\FPI\resources\sample\workspace\CobolSample.fpi

XML Record (Middleware Type: Cobol Copybook)

The following screen shot shows part of the application window when a Transaction Record attribute is selected in the tree:


Financial Process Integrator
Workspace Edit Window Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions

Transaction Records
XML: CobolXML_3
Field: Card_Number
Group: Account_Info
Field: Account_Number
Field: Account_Name
Group: Address_Details
Group: Account_Info_All

Field Description: This returns the Account Name
Field Occurrences: 1
Field Name: Account_Name
Field Type: Double
Length: 8
Field Encoding: ASCII
Field Padding: 0
Dec Before:
Dec After:
Field Signed: False

Currently Selected Workspace is: D:\eontec\FPI\resources\sample\workspace\XMLSample.fpi

13 Financial Components

The Components list is found by selecting the middle tab in the application window, titled ‘Imported Siebel Components’. The Siebel Components node is a placeholder list for all Financial Objects and Financial Components that have been imported into your workspace. Beneath the Siebel Components node, there will be a list of nodes for Financial Objects and Financial Components, which will contain a list of zero or more Financial Objects and Components that have been previously imported. A user must import Siebel Components from an XML file format so they can be used in mapping host transactions.

The Financial Component list is added beneath the Siebel Components node. The Financial Component node is a placeholder list for all Financial Components that exist in your workspace. Beneath the Financial Component node, there will be a list of zero or more Financial Components that have been previously imported. These are denoted by a “Financial Component:” label in front of the name. A Financial Component represents a session based java component that is being used by your application. Financial Components are imported from the same XML file that your model contains. At present, Financial Components are not used in the mapping of transaction metadata, so this list will often be empty.

13.1 Financial Component Operations

The following operations are available to the user when the Siebel Components list node is selected in the tree:

13.1.1 Importing a Siebel Model

Selecting the Import Siebel model menu item will launch a wizard that allows you to import a representation of a Siebel Design Model. Siebel Models are imported in XML file format that represents an application’s Automated Methodology Design model, which is generated from the Siebel design tools. The Financial Process Integrator automatically parses the business objects in the model, and stores its associated properties and structure in the workspace. Individual attributes of the business object are displayed as new nodes, listed below the business object node. Each attribute has properties associated with it, which were automatically parsed from the XML definition of the business object when it is imported

13.1.2 Removing a Siebel Model

Selecting the Remove a Siebel Model menu item will remove all previously imported objects and processes from your workspace. If a business object that is part of a user defined host transaction is removed, you will have to go back and re-map the fields in that host transaction to ensure that it is mapped using the most recent business object definitions in your workspace.

13.2 Financial Objects

The Financial Objects list is automatically added beneath the Siebel Components node. The Financial Objects node is a placeholder list for all Financial Objects that have been imported into your workspace. Beneath the Financial Objects node there will be a list of zero or more Financial Objects that have been previously imported. These are denoted by a “Financial Object:” label in front of the name. A Financial Object represents an entity based java component that is being used by your application. A user must import financial objects from an XML file format so they can be used in mapping host transactions. The following menu items (actions) are available to the user when the Financial Objects list node is selected in the tree:

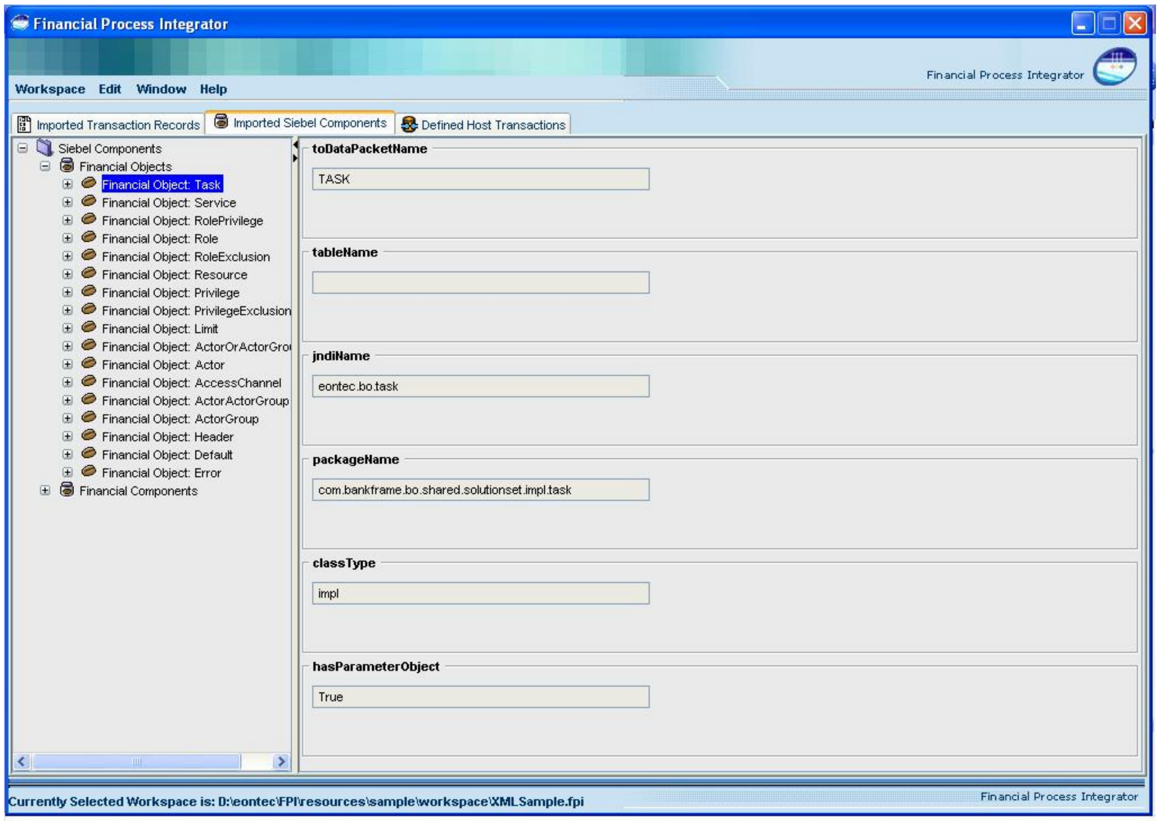
13.3 Financial Component Properties

The Financial Object list has no properties, but each Financial Object as well as its attributes and operators will have properties associated with it, which can be viewed in the right window of the FPI when one is selected. These properties were parsed from the XML file when the Import Financial Object wizard was finished. Properties on a financial object are not editable and must be changed in the design tools. Each Financial Object node has some of the following properties associated with it:

Package Name	The java package name that this Financial Object is in.
Class Type	Class Type (Domain, Sector etc.). Only implementation level classes are displayed in the Host Tool.
JNDI Name	The Java Naming and Directory Interface name that this Financial Object is assigned.
Table Name	The relational database table that this object is associated with. (CMP scenario)
ToDataPacket Name	The `DataPacket` name that this Financial Object is associated with.

All Financial Objects (attributes and operators also) have a number of properties associated with them. These properties are parsed from the models generated from the design tools, and are displayed in the right window when the attribute is selected.

The following screen shot shows part of the application window when a Business Object node is selected, from a list of many objects that were imported:


Financial Process Integrator
Workspace Edit Window Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Siebel Components
Financial Objects
Financial Object: Task
Financial Object: Service
Financial Object: RolePrivilege
Financial Object: Role
Financial Object: RoleExclusion
Financial Object: Resource
Financial Object: Privilege
Financial Object: PrivilegeExclusion
Financial Object: Limit
Financial Object: ActorOrActorGro
Financial Object: Actor
Financial Object: AccessChannel
Financial Object: ActorActorGroup
Financial Object: ActorGroup
Financial Object: Header
Financial Object: Default
Financial Object: Error
Financial Components

toDataPacketName: TASK

tableName:

jndiName: eontec.bo.task

packageName: com.bankframe.bo.shared.solutionset.impl.task

classType: impl

hasParameterObject: True

Currently Selected Workspace is: D:\eontec\FPI\resources\sample\workspace\XMLSample.fpi

14 Cobol Copybook Example

Now that we've installed and had a brief overview of the Financial Process Integrator tool, we will walk though a simple example. This example uses Cobol Copybooks as the transaction record type. In general there's going to be one copybook for the request and one for the response. They can be the same but for the sake of clarity let's separate them. The request copybook, usually denoted by `MID`, looks like the following:

14.1 Importing Cobol Copybooks

```
000100*****************************************************************
000200**  SAMPLE COPYBOOK  - MID-CUSTFINDBYPK.txt
000300*****************************************************************
000400 01 MID-CUSTFINDBYPK.
000500        05 MID-CUSTFINDBYPK-CUSTOMERID       PIC 9(10).
```

Without going into too much detail into the structure of a copybook, this copybook has one field: '`MID-CUSTFINDBYPK-CUSTOMERID`' that can be 'mapped' to an entity bean field. It is a number typed field, denoted by `PIC 9` of length ten, denoted by the number in braces following the field definition. Now let's see the response copybook, or `MOD`:

```
000010*****************************************************************
000020**  SAMPLE COPYBOOK  - MOD-CUSTFINDBYPK.txt
000030*****************************************************************
000100 01 MOD-CUSTFINDBYPK.
000500   03 MOD-CUSTFINDBYPK-ERR-NO                  PIC 9(10).
000500   03 MOD-CUSTFINDBYPK-ERR-MSG                 PIC X(10).
000500   03 MOD-CUSTFINDBYPK-C-FIRST-NAME            PIC X(20).
000510   03 MOD-CUSTFINDBYPK-C-LAST-NAME             PIC X(20).
000520   03 MOD-CUSTFINDBYPK-A-POST-CODE             PIC X(15).
000530   03 MOD-CUSTFINDBYPK-A-LINE-1               PIC X(20).
000540   03 MOD-CUSTFINDBYPK-A-LINE-2               PIC X(20).
000550   03 MOD-CUSTFINDBYPK-A-LINE-3               PIC X(20).
000570   03 MOD-CUSTFINDBYPK-A-COUNTRY              PIC X(20).
```

We can now import the copybooks into the FPI. Go to the EDIT menu and select the Import | Transaction Record entry in the menu. When the wizard appears, choose Cobol Copybook as the record type to import and hit next. On the next panel, browse through the file system to find your copybook `MID` file and press finish. It should be in the `<%FPI Installation%>/resources/sample/copybookrecords` directory, named MID-CUSTFINDBYPK.txt. You should now see a new entry in your transaction records list if you expand the root node. By further expanding the sub-nodes of the transaction record node you will see any groups and fields within your copybook. You should check these to ensure that the import was correctly accomplished. You can now go ahead and import the other `MOD` copybook. It is called `MOD-CUSTFINDBYPK.txt` and can be found in the same directory as above. Repeat the same steps to import this record.

14.2 Importing Financial Objects

‘Financial Object’ is another term for an Enterprise Java Entity Bean. To import a financial object, select the ‘Financial Objects’ node in the tree. Once again, go to the Edit menu and select Import | Siebel Model. When the wizard appears, click the browse button and browse for your Model file. This should have a `.xml` extension. There are two different models shipped with the tool. The first is the Branch Teller model and the second is the Entitlements model. They can be found in the `<%FPI Installation%>/resources/sample/SiebelModel` directory.

Now expand the ‘Siebel Components’ and ‘Financial Objects’ nodes. You should see all nine financial objects. You can expand each business object node as well and see each objects attributes.

14.3 Creating Host Systems and Subsystems

Before we can create a host transaction we need to create a host subsystem in which it resides and a host system which the host subsystem in turn resides. On the third tab, titled ‘Defined Host Transactions’ Select the ‘Host Systems’ node and add a host system by selecting the Edit menu and choosing the New Host System item. (This could alternatively be accomplished by selecting the Host Systems node in the tree, right clicking on it and choosing the Add Host System item from the pop-up menu). You will be presented with an add host system wizard. Fill in ‘zSeries’ as the host system name and click finish.
You will see the new host system in the ‘Host Systems’ node. Now select the ‘zSeries’ node you just created. Adding a host subsystem is much the same as a host system. It’s left as an exercise to the reader to create a host subsystem within the host system ‘zSeries’ named ‘Branch’.
We’re almost ready to add a host transaction but first we have to import a host connector.

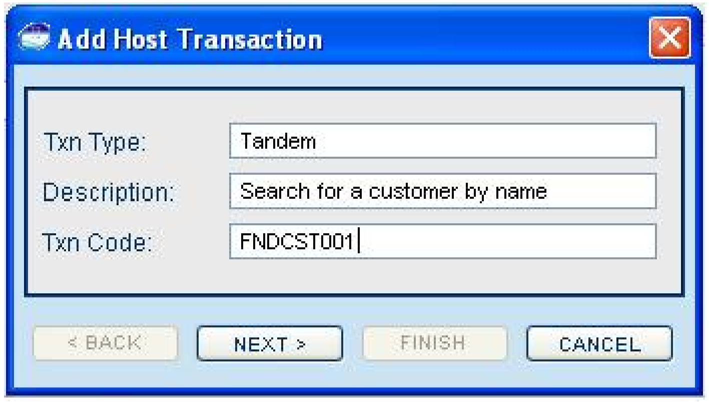
14.4 Importing Host Connectors

A host connector is a Java class that communicates with the piece of software (middleware) that connects your host system backend to the MCA Financial Process Integrator. The MCA Financial Process Integrator needs certain properties to hand to the connector to connect to the middleware. Host connectors are defined in the MCA properties file named `BankframeResource.properties`, which should reside in the directory where an instance of MCA Services is installed. You might also find this file in the repository. Click on the

Edit menu and select the Import -> Host Connectors menu item. This will launch a wizard that prompts a user to brows for the properties file mentioned above. For the purposes of this sample, a properties file is provided in the `<%FPI Installation%>/resources/sample` directory. Browse for this file, and then click finish on the wizard. Notice that one or more connectors will now be visible underneath the System node of the tree.

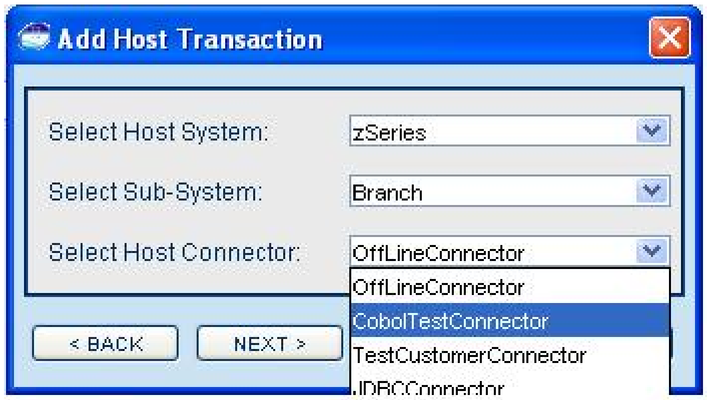
14.5 Creating a Host Transaction

The host transaction is the definition of a transaction between the host system and a client. Transaction Records (in this example – Copybooks) define the inputs and outputs of this transaction, as we'll see later on. To add a host transaction, select the 'Branch' node you created in a previous section and right click to choose the Add Host Transaction menu item. You will be presented with an add host transaction wizard. Enter the details the same as in the dialogue box below:


Add Host Transaction

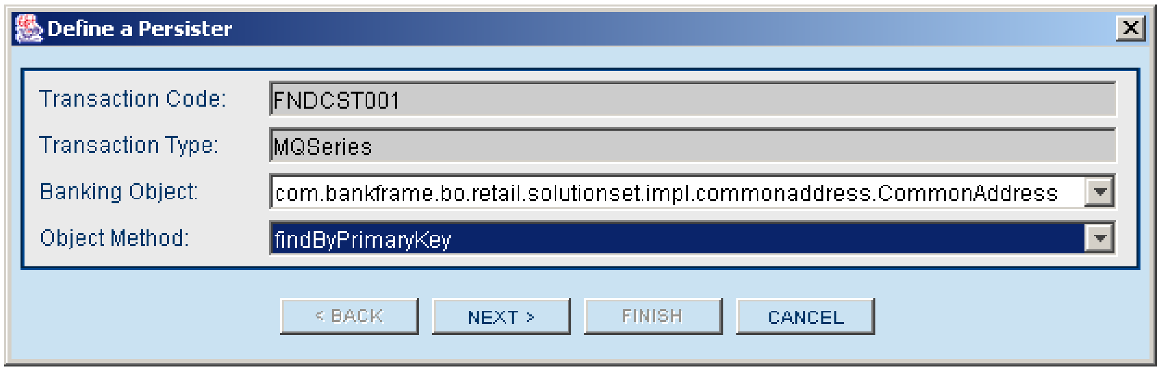
Txn Type: Tandem
Description: Search for a customer by name
Txn Code: FNDCST001

and press 'Next'. In the next panel, select the System and SubSystem that you created previsouly, and choose the Host Connector from the drop-down lists provided. It should look something like this:


Add Host Transaction

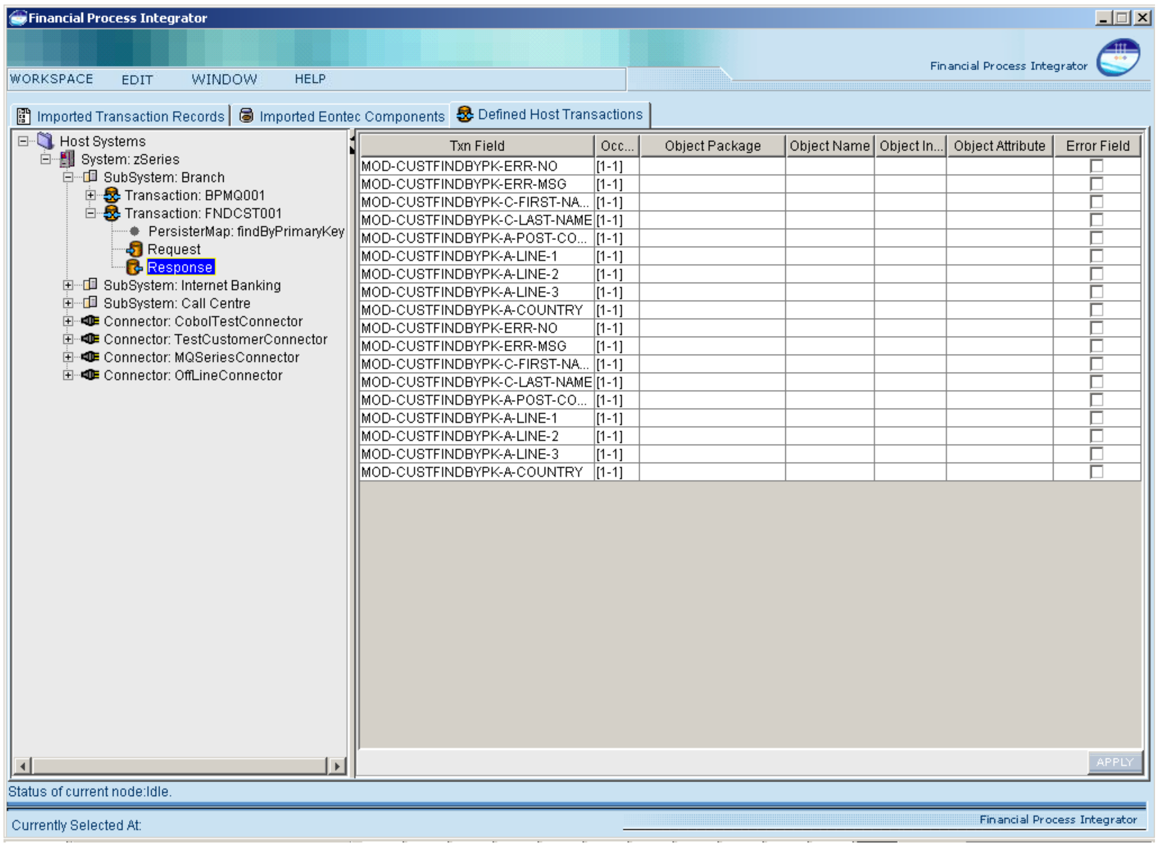
Select Host System: zSeries
Select Sub-System: Branch
Select Host Connector: OffLineConnector

Then press 'Next'. In this last panel select your Request record as `MID-CUSTFINDBYPK` and your Response record as `MOD-CUSTFINDBYPK`. Now press finish to create your host transaction. You should see a host transaction under 'SubSystem:Branch' with the name `FNDCST001`. If you select that node and expand it there will be two mapping groups under this node named Request and Response. Next we will define a Persister for this transaction. If you right click on the `FNDCST001` transaction node, and click on the Define Persister menu item it will launch a wizard. The transaction code and type values are filled in automatically for you. From the drop-down lists, select the `CommonAddress` class and the `findByPrimaryKey` method. (You may have to expand the wizard panel horizontally to see the full package name list). Your wizard should look something like the following screen shot:



Transaction Code:	FNDCST001
Transaction Type:	MQSeries
Banking Object:	com.bankframe.bo.retail.solutionset.impl.commonaddress.CommonAddress
Object Method:	findByPrimaryKey

Click the Next button. In the second panel choose a cache mechanism as `MEMORY` and set the timeout value to `500` milliseconds. There should now be a Persister node underneath the Host Transaction node, at the same level as your Request and Response. If you select the Response node, in the window to the right you should now see a table with numerous rows. It should look something like this:


Financial Process Integrator
WORKSPACE
EDIT
WINDOW
HELP
Financial Process Integrator
Imported Transaction Records
Imported Eontec Components
Defined Host Transactions
APPLY
Financial Process Integrator

Host Systems
- System: zSeries
  - SubSystem: Branch
    - Transaction: BPMQ001
    - Transaction: FNDCST001
      - PersisterMap: findByPrimaryKey
      - Request
      - Response
  - SubSystem: Internet Banking
  - SubSystem: Call Centre
  - Connector: CobolTestConnector
  - Connector: TestCustomerConnector
  - Connector: MQSeriesConnector
  - Connector: OffLineConnector

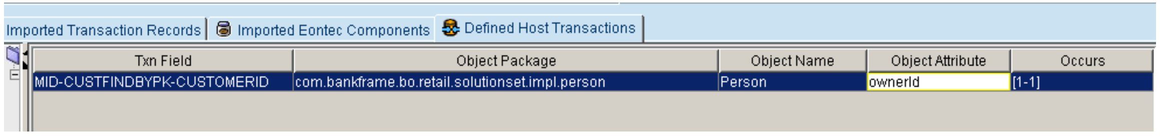
Txn Field	Occ...	Object Package	Object Name	Object In...	Object Attribute	Error Field
MOD-CUSTFINDBYPK-ERR-NO	[1-1]					☐
MOD-CUSTFINDBYPK-ERR-MSG	[1-1]					☐
MOD-CUSTFINDBYPK-C-FIRST-NA...	[1-1]					☐
MOD-CUSTFINDBYPK-C-LAST-NAME	[1-1]					☐
MOD-CUSTFINDBYPK-A-POST-CO...	[1-1]					☐
MOD-CUSTFINDBYPK-A-LINE-1	[1-1]					☐
MOD-CUSTFINDBYPK-A-LINE-2	[1-1]					☐
MOD-CUSTFINDBYPK-A-LINE-3	[1-1]					☐
MOD-CUSTFINDBYPK-A-COUNTRY	[1-1]					☐
MOD-CUSTFINDBYPK-ERR-NO	[1-1]					☐
MOD-CUSTFINDBYPK-ERR-MSG	[1-1]					☐
MOD-CUSTFINDBYPK-C-FIRST-NA...	[1-1]					☐
MOD-CUSTFINDBYPK-C-LAST-NAME	[1-1]					☐
MOD-CUSTFINDBYPK-A-POST-CO...	[1-1]					☐
MOD-CUSTFINDBYPK-A-LINE-1	[1-1]					☐
MOD-CUSTFINDBYPK-A-LINE-2	[1-1]					☐
MOD-CUSTFINDBYPK-A-LINE-3	[1-1]					☐
MOD-CUSTFINDBYPK-A-COUNTRY	[1-1]					☐

Status of current node:Idle.

Currently Selected At:

14.6 Mapping Fields to Attributes

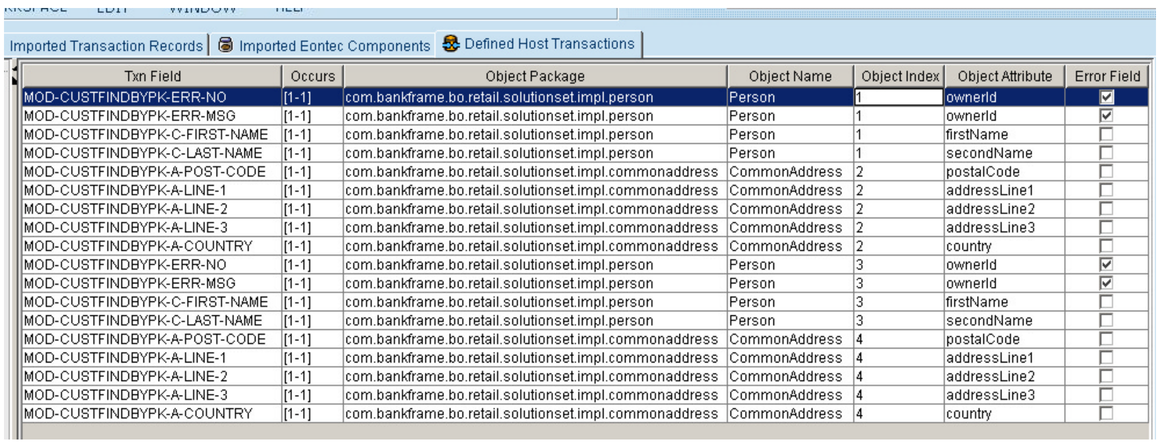
This is the heart of the FPI tool. Each row represents one **possible occurrence** (see appendix) of a field in an instance of a copybook. To map an occurrence to a financial object attribute, first select the package the target bean is in by selecting it in the second column drop down box. Once you do this the next column, the financial object name column, will be populated. Now select the desired financial object name, thereby populating the financial object attribute column. Finally select the financial object attribute name. Note that no change will be made in the database until you commit these changes by pressing the 'Apply' button. Let's go ahead and map the Request field (note there is only one in the request record) in the following way:


Imported Transaction Records
Imported Eontec Components
Defined Host Transactions

Txn Field	Object Package	Object Name	Object Attribute	Occurs
MID-CUSTFINDBYPK-CUSTOMERID	com.bankframe.bo.retail.solutionset.impl.person	Person	ownerId	[1-1]

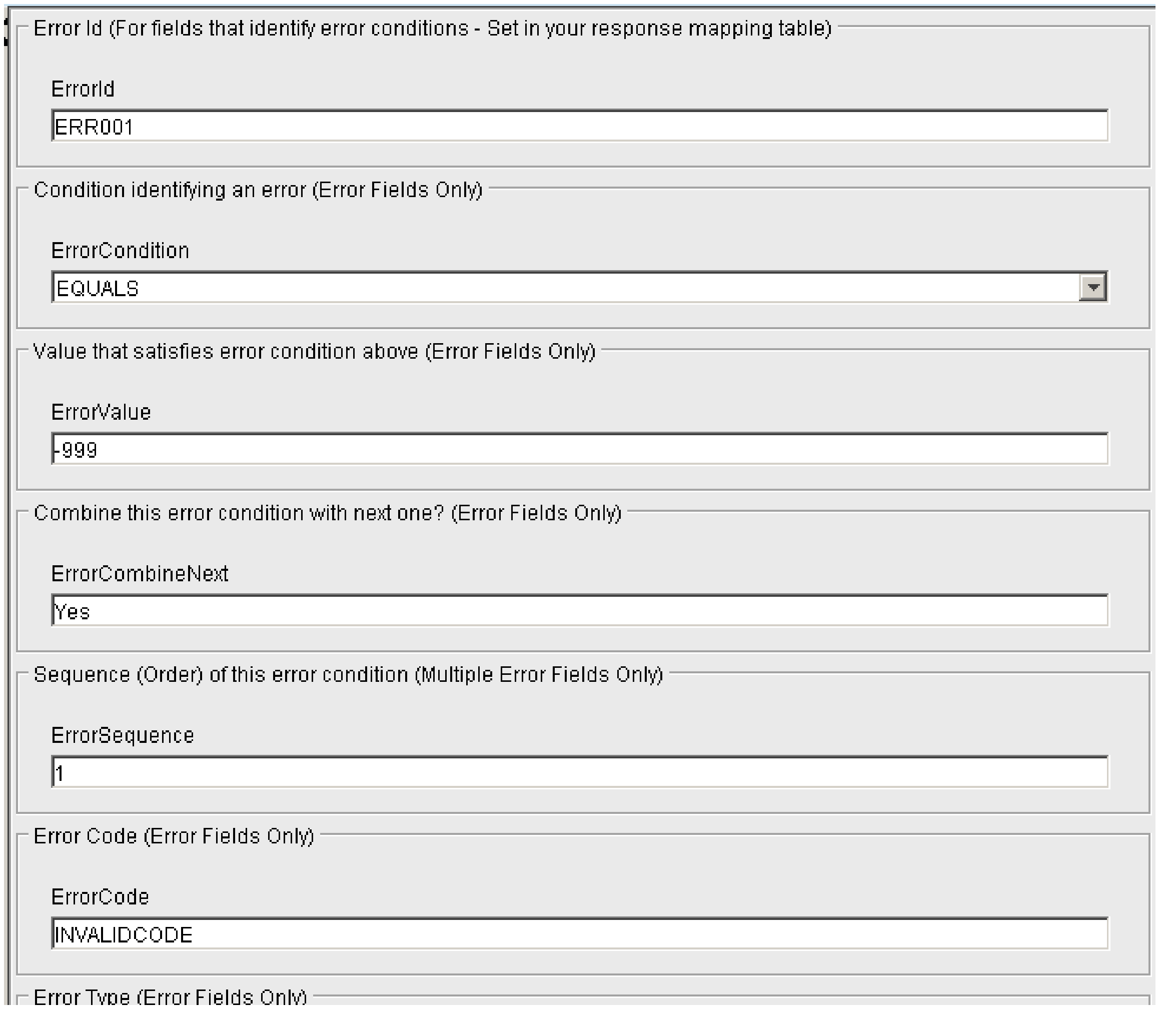
You will probably want to expand the divider between the window panes so that you have more room to work with the mapping table. You may also have to resize some of the columns to see all of the values contained in the selection lists. Don't forget to hit the 'Apply' button when finished.

Now complete the Response mappings like in the following screen shot:


Imported Transaction Records
Imported Eontec Components
Defined Host Transactions

Txn Field	Occurs	Object Package	Object Name	Object Index	Object Attribute	Error Field
MOD-CUSTFINDBYPK-ERR-NO	[1-1]	com.bankframe.bo.retail.solutionset.impl.person	Person	1	ownerId	☑
MOD-CUSTFINDBYPK-ERR-MSG	[1-1]	com.bankframe.bo.retail.solutionset.impl.person	Person	1	ownerId	☑
MOD-CUSTFINDBYPK-C-FIRST-NAME	[1-1]	com.bankframe.bo.retail.solutionset.impl.person	Person	1	firstName	☐
MOD-CUSTFINDBYPK-C-LAST-NAME	[1-1]	com.bankframe.bo.retail.solutionset.impl.person	Person	1	secondName	☐
MOD-CUSTFINDBYPK-A-POST-CODE	[1-1]	com.bankframe.bo.retail.solutionset.impl.commonaddress	CommonAddress	2	postalCode	☐
MOD-CUSTFINDBYPK-A-LINE-1	[1-1]	com.bankframe.bo.retail.solutionset.impl.commonaddress	CommonAddress	2	addressLine1	☐
MOD-CUSTFINDBYPK-A-LINE-2	[1-1]	com.bankframe.bo.retail.solutionset.impl.commonaddress	CommonAddress	2	addressLine2	☐
MOD-CUSTFINDBYPK-A-LINE-3	[1-1]	com.bankframe.bo.retail.solutionset.impl.commonaddress	CommonAddress	2	addressLine3	☐
MOD-CUSTFINDBYPK-A-COUNTRY	[1-1]	com.bankframe.bo.retail.solutionset.impl.commonaddress	CommonAddress	2	country	☐
MOD-CUSTFINDBYPK-ERR-NO	[1-1]	com.bankframe.bo.retail.solutionset.impl.person	Person	3	ownerId	☑
MOD-CUSTFINDBYPK-ERR-MSG	[1-1]	com.bankframe.bo.retail.solutionset.impl.person	Person	3	ownerId	☑
MOD-CUSTFINDBYPK-C-FIRST-NAME	[1-1]	com.bankframe.bo.retail.solutionset.impl.person	Person	3	firstName	☐
MOD-CUSTFINDBYPK-C-LAST-NAME	[1-1]	com.bankframe.bo.retail.solutionset.impl.person	Person	3	secondName	☐
MOD-CUSTFINDBYPK-A-POST-CODE	[1-1]	com.bankframe.bo.retail.solutionset.impl.commonaddress	CommonAddress	4	postalCode	☐
MOD-CUSTFINDBYPK-A-LINE-1	[1-1]	com.bankframe.bo.retail.solutionset.impl.commonaddress	CommonAddress	4	addressLine1	☐
MOD-CUSTFINDBYPK-A-LINE-2	[1-1]	com.bankframe.bo.retail.solutionset.impl.commonaddress	CommonAddress	4	addressLine2	☐
MOD-CUSTFINDBYPK-A-LINE-3	[1-1]	com.bankframe.bo.retail.solutionset.impl.commonaddress	CommonAddress	4	addressLine3	☐
MOD-CUSTFINDBYPK-A-COUNTRY	[1-1]	com.bankframe.bo.retail.solutionset.impl.commonaddress	CommonAddress	4	country	☐

Remember to click on the Apply button once you have filled in all the fields. Alternatively, you can right click on the 'Response' node and select the 'Apply Changes' menu item from the pop up menu.
You'll notice that some of the fields in the mapping are marked as Error Fields via the check-box in the last column of the table. This column appears for Cobol Copybook Response mappings only, and indicates that this field should be checked for error conditions when it is returned from the Host system. You must set error conditions on the Transaction Record that you imported under the 'Imported Transaction Records' Tab. Each field from an imported copybook has error fields that a user can edit, which populates the `RESPONSE_ERROR_CONDITION` metadata table. These fields look like the following:



Error Id (For fields that identify error conditions - Set in your response mapping table)

ErrorId

ERR001

Condition identifying an error (Error Fields Only)

ErrorCondition

EQUALS

Value that satisfies error condition above (Error Fields Only)

ErrorValue

-999

Combine this error condition with next one? (Error Fields Only)

ErrorCombineNext

Yes

Sequence (Order) of this error condition (Multiple Error Fields Only)

ErrorSequence

1

Error Code (Error Fields Only)

ErrorCode

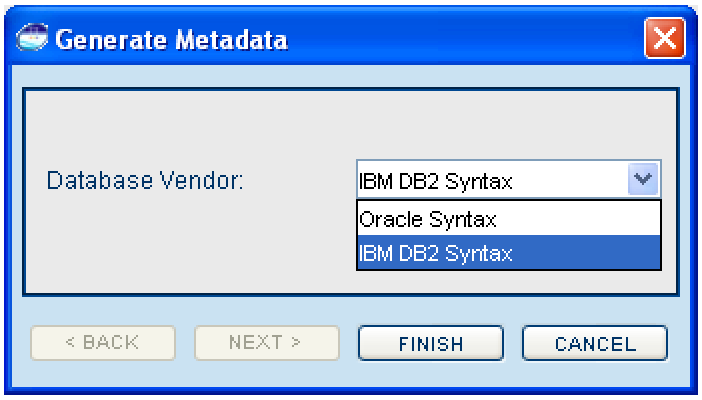
INVALIDCODE

Error Type (Error Fields Only)

A user can edit these fields, then press the 'Apply' button to set error conditions. Now we're ready to generate the SQL Insert statements for the metadata required by the MCA Financial Process Integrator.

14.7 Generating Metadata

Click on the Workspace menu item in the application window, and select the 'Generate SQL' menu item. This will bring up a wizard that prompts you to choose a metadata format, and a database vendor to generate the metadata.

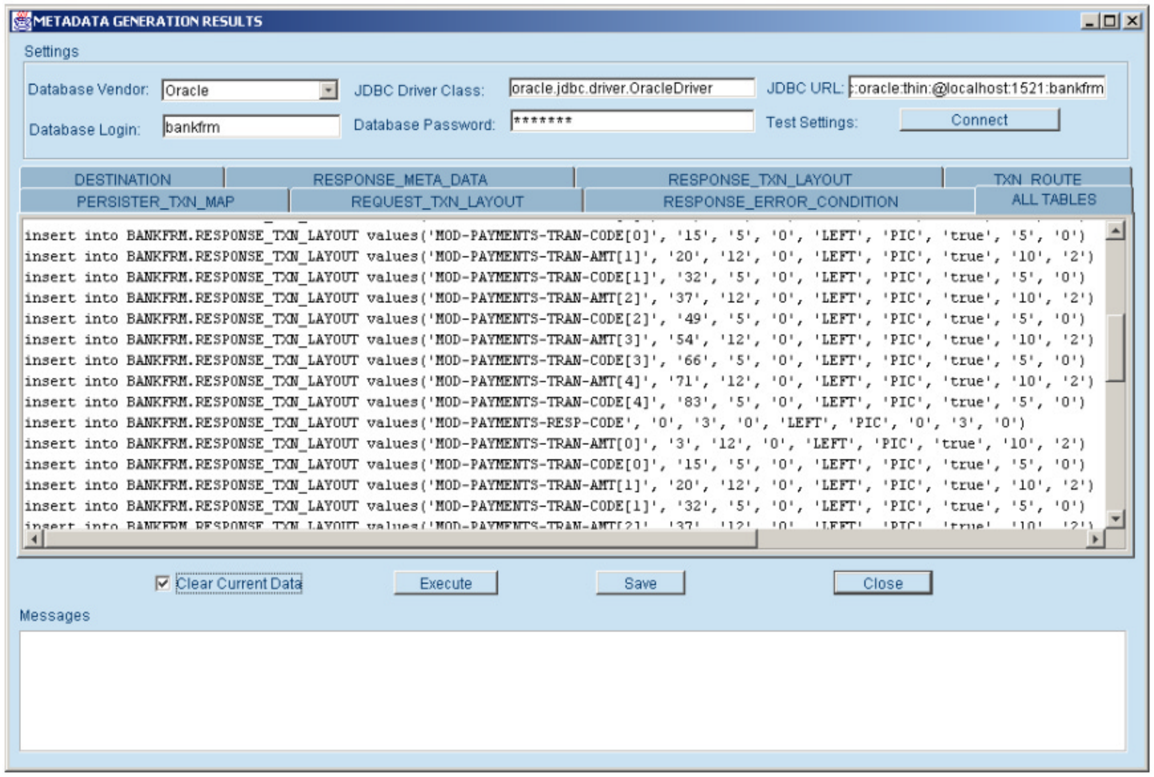


Database Vendor: IBM DB2 Syntax

Oracle Syntax
IBM DB2 Syntax

The Metadata schema formats are dependent on the database vendor option selected.

This will generate SQL insert statements based on the transactions, mappings and values you have defined within the Financial Process Integrator. It will bring up a panel displaying individual table values required for the Metadata format you selected. Each tab in the new window will have the required SQL Insert statements for various metadata tables, plus one tab representing all tables. You should see the resultant SQL in a pop up window like the following:



Database Vendor: Oracle
JDBC Driver Class: oracle.jdbc.driver.OracleDriver
JDBC URL: :oracle:thin:@localhost:1521:bankfrm
Database Login: bankfrm
Database Password: *******
Test Settings:

```
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[0]', '15', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[1]', '20', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[1]', '32', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[2]', '37', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[2]', '49', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[3]', '54', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[3]', '66', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[4]', '71', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[4]', '83', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-RESP-CODE', '0', '3', '0', 'LEFT', 'PIC', '0', '3', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[0]', '3', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[0]', '15', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[1]', '20', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[1]', '32', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[2]', '37', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
```

[✓ Clear Current Data]

The section at the top of the window can be used if you wish to connect to a relational database system and execute your SQL statements from the Integrator. Once you have filled in the proper settings for the database system you wish to connect to, pressing the Connect button will test to see if a connection can be made. If the connection test is successful then you can press the Execute button, which is below the insert statements, to update the database immediately. By checking the Clear Current Data checkbox this ensures that any existing data will first be deleted from your database. Only the six tables DESTINATION, RESPONSE_META_DATA, RESPONSE_TXN_LAYOUT, TXN_ROUTE, PERSISTER_TXN_MAP, REQUEST_TXN_LAYOUT and RESPONSE_ERROR_CONDITION will be cleared of data. Any error messages or status updates from the attempt to execute will appear in the Messages text area at the bottom of the window. In addition, you also have the option to save the SQL file to your hard disk for later insertion and execution into your database tables.

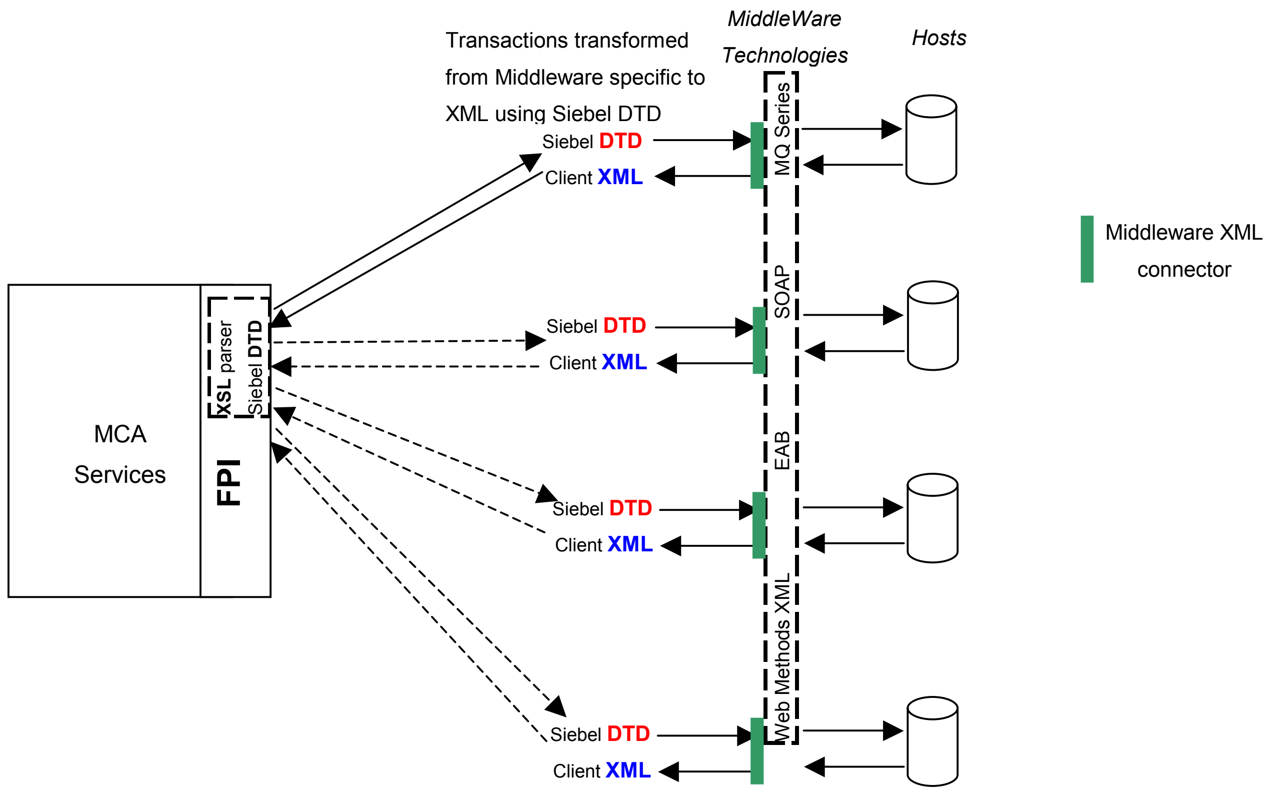
Financial Process Integrator settings that are configured in the BankFrameConstants.properties file

Refer to the appendix on FPI Configuration for a description of

15 XML Transaction Record Example

In this example we are going to be importing an XML Transaction Record. The underlying middleware type of this host transacton will be cobol.

15.1 Architecture Overview for XML Transaction Record



Below is an example of expressing a cobol copybook via the XML Transaction Record.

```
<?xml version="1.0"?>
<!DOCTYPE Transaction SYSTEM "file:///C://temp//dtds//FPITransaction.dtd">
<Transaction hostMiddleWare="cobol">
        <TransactionOverview>This transaction returns Account and Address information</TransactionOverview>
        <Field fieldOccurrences="4" fieldName="Card_Number" fieldType="Long" length="8" javaClass=""
javaMethod="" fieldPadding="0" decBefore="" decAfter="" fieldSigned="False" fieldAligned="LEFT" mandatory="Yes">
                <FieldOverview>This returns the Card Number</FieldOverview>
        </Field>
        <Group groupName="Account_Info" groupOccurrences="2" redefines="">
                <GroupOverview>This returns the Account information</GroupOverview>
                <Field fieldOccurrences="1" fieldName="Account_Number" fieldType="Double" length="8"
                javaClass="" javaMethod="" fieldEncoding="ASCII" fieldPadding="0" decBefore="" decAfter=""
                fieldSigned="False" fieldAligned="LEFT" mandatory="Yes">
                <FieldOverview>This returns the Account Number</FieldOverview>
                </Field>
                <Field fieldOccurrences="1" fieldName="Account_Name" fieldType="Double" length="8" javaClass=""
                javaMethod="" fieldEncoding="ASCII" fieldPadding="0" decBefore="" decAfter="" fieldSigned="False"
                fieldAligned="RIGHT" mandatory="Yes">
                <FieldOverview>This returns the Account Name</FieldOverview>
                </Field>
        </Group>
        <Group groupName="Address_Details" groupOccurrences="2" redefines="">
                <GroupOverview>This returns the Account information</GroupOverview>
                <Field fieldOccurrences="3" fieldName="Street_Address" fieldType="String" length="8" javaClass=""
                javaMethod="" fieldEncoding="ASCII" fieldPadding="0" decBefore="" decAfter="" fieldSigned="False"
                fieldAligned="LEFT" mandatory="Yes">
                <FieldOverview>This returns the Street Address</FieldOverview>
                </Field>
                <Field fieldOccurrences="1" fieldName="State" fieldType="String" length="8" javaClass=""
                javaMethod="" fieldEncoding="ASCII" fieldPadding="0" decBefore="" decAfter="" fieldSigned="False"
                fieldAligned="RIGHT" mandatory="Yes">
                <FieldOverview>This returns the State</FieldOverview>
                </Field>
                <Field fieldOccurrences="1" fieldName="Postcode" fieldType="String" length="8" javaClass=""
                javaMethod="" fieldEncoding="ASCII" fieldPadding="0" decBefore="" decAfter="" fieldSigned="False"
                fieldAligned="RIGHT" mandatory="Yes">
                <FieldOverview>This returns the Postcode</FieldOverview>
                </Field>
        </Group>
        <Group groupName="Account_Info_All" groupOccurrences="1" redefines="">
                <Group groupName="Account_Info_All_1" groupOccurrences="1" redefines="Account_Info">
                        <GroupOverview>This returns the Account information for Type 1 in a different
                        format</GroupOverview>
                        <Field fieldOccurrences="1" fieldName="Account_Number_Four_Digits_1"
                        fieldType="Long" length="8" javaClass="" javaMethod="" fieldEncoding="ASCII"
                        fieldPadding="0" decBefore="" decAfter="" fieldSigned="False" fieldAligned="LEFT"
                        mandatory="Yes">
                        <FieldOverview>This returns the Account number as four digits for type 1
                        account</FieldOverview>
                        </Field>
                        <Group groupName="Account_Info_All_2" groupOccurrences="1"
                        redefines="Account_Info">
                        <GroupOverview>This returns the Account information in a different
                        format</GroupOverview>
                                <Field fieldOccurrences="1" fieldName="Account_Number_Four_Digits"
                                fieldType="Long" length="8" javaClass="" javaMethod="" fieldEncoding="ASCII"
                                fieldPadding="0" decBefore="" decAfter="" fieldSigned="False"
                                fieldAligned="LEFT" mandatory="Yes">
                                <FieldOverview>This returns the Account number as four digits</FieldOverview>
```

```
                    </Field>
                    <Field fieldOccurrences="1" fieldName="Account_Number_Eight_Digits"
                    fieldType="Long" length="8" javaClass="" javaMethod="" fieldEncoding="ASCII"
                    fieldPadding="0" decBefore="" decAfter="" fieldSigned="False"
                    fieldAligned="LEFT" mandatory="Yes">
                    <FieldOverview>This returns the Account number as eight
                    digits</FieldOverview>
                    </Field>
               </Group>
          </Group>
     </Group>
</Transaction>
```

15.2 Importing Financial Objects

'Financial Object' is another term for an Enterprise Java Entity Bean. To import a financial object, select the 'Financial Objects' node in the tree, select Import | Siebel Model from the Edit menu. When the wizard appears, click the browse button and browse for the Model file. This should have a `.xml` extension. There are two different models shipped with the tool. The first is the Branch Teller model and the second is the Entitlements model. They can be found in the `<%FPI Installation%>/ resources/sample/SiebelModel` directory.

Now expand the 'Siebel Components' and 'Financial Objects' nodes. You should see all nine financial objects. You can expand each business object node as well and see each objects attributes.

15.3 Creating Host Systems and Subsystems

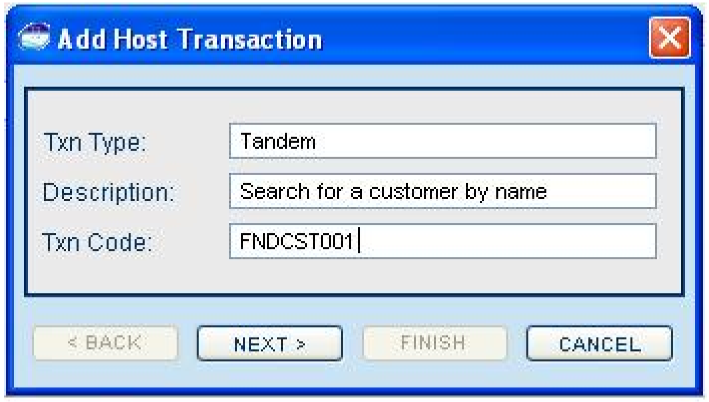
Before we can create a host transaction we need to create a host subsystem in which it resides and a host system which the host subsystem in turn resides. On the third tab, titled 'Defined Host Transactions' select the 'Host Systems' node and add a host system by selecting the Edit menu and choosing the New Host System item. (This could alternatively be accomplished by selecting the Host System node in the tree, right clicking on it and choosing the Add Host System item from the pop-up menu). You will be presented with an add host system wizard. Fill in 'zSeries' as the host system name and click finish.
You will see the new host system in the 'Host Systems' node. Now select the 'zSeries' node you just created. Adding a host subsystem is much the same as a host system, create a host subsystem within the host system 'zSeries' named 'Branch'.
We're almost ready to add a host transaction but first we have to import a host connector.

15.4 Importing Host Connectors

A host connector is a Java class that communicates with the piece of software (middleware) that connects your host system backend to the MCA Services Financial Process Integrator. The MCA Services Financial Process Integrator needs certain properties to pass to the connector to connect to the middleware. Host connectors are defined in the MCA properties file named `BankframeResource.properties`, which should reside in the directory where an instance of MCA Services is installed. You might also find this file in the repository. Click on the Edit menu and select the Import -> Host Connectors menu item. This will launch a wizard that prompts a user to browse for the properties file mentioned above. For the purpose of this example, a properties file is provided in the `<%FPI Installation%>/resources/sample` directory. Browse for this file, and then click finish on the wizard. Notice that one or more connectors will now be visible underneath the System node of the tree.

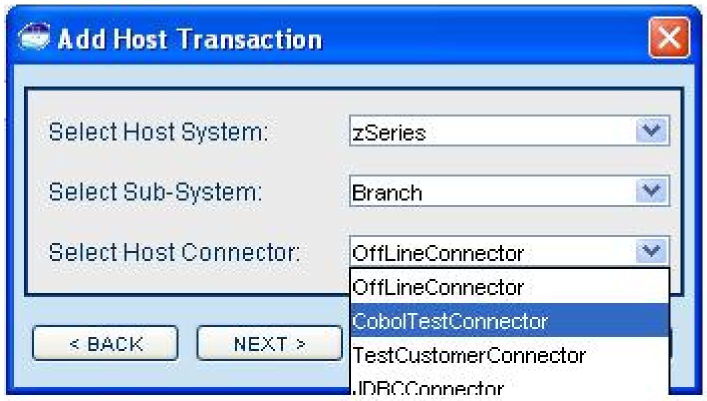
15.5 Creating a Host Transaction

The host transaction is the definition of a transaction between the host system and a client. Transaction Records (in this example – Copybooks) define the inputs and outputs of this transaction. To add a host transaction, select the 'Branch' node you created previously and right click to choose the Add Host Transaction menu item. You will be presented with an add host transaction wizard. Enter the details the same as in the dialogue box below:



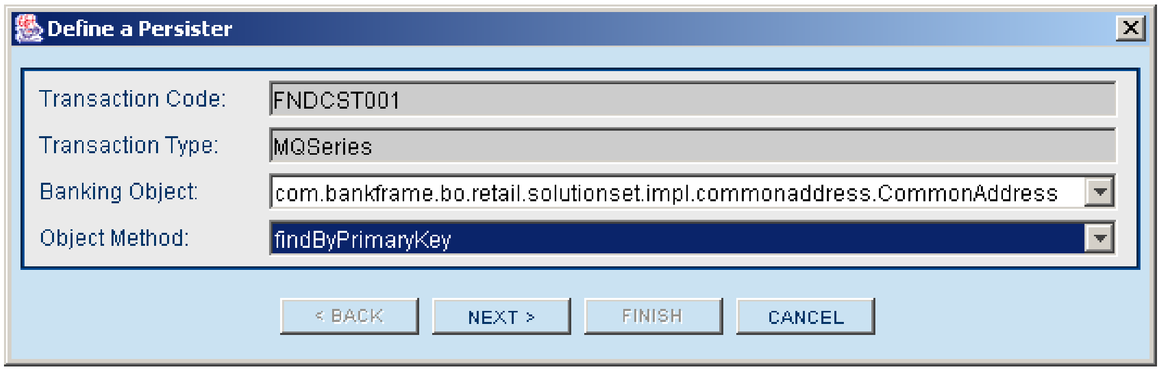
Txn Type: Tandem
Description: Search for a customer by name
Txn Code: FNDCST001

and select 'Next'. In the next panel, select the System and SubSystem that you created previsouly, and choose the Host Connector from the drop-down list provided. It should look something like this:



Select Host System: zSeries
Select Sub-System: Branch
Select Host Connector: OffLineConnector

Then press 'Next'. In this last panel select your Request record as `MID-CUSTFINDBYPK` and your Response record as `MOD-CUSTFINDBYPK`. Now press finish to create your host transaction. You should see a host transaction under 'SubSystem:Branch' with the name `FNDCST001`. If you select that node and expand it there will be two mapping groups under this node named Request and Response. Next we will define a Persister for this transaction. If you right click on the `FNDCST001` transaction node, and click on the Define Persister menu item it will launch a wizard. The transaction code and type values are filled in automatically for you. From the drop-down lists, select the `CommonAddress` class and the `findByPrimaryKey` method. (You may have to expand the wizard panel horizontally to see the full package name list). Your wizard should look something like the following screen shot:

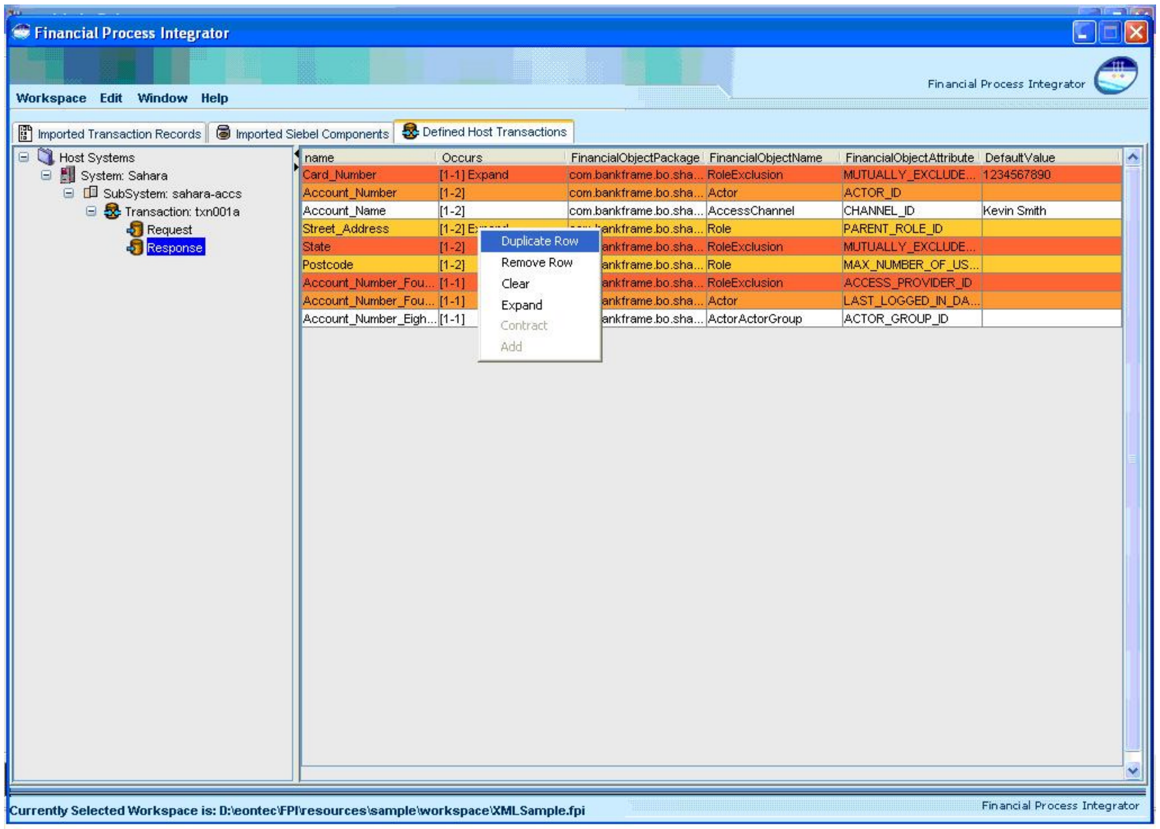


Transaction Code:	FNDCST001
Transaction Type:	MQSeries
Banking Object:	com.bankframe.bo.retail.solutionset.impl.commonaddress.CommonAddress
Object Method:	findByPrimaryKey

Click the Next button. In the second panel choose a cache mechanism as `MEMORY` and set the timeout value to `500` milliseconds. There should now be a Persister node underneath the Host Transaction node, at the same level as your Request and Response. If you select the Response node, in the window to the right you should now see a table with numerous rows.

15.6 Mapping Fields to Attributes

As described in section 9 the user can now map the transaction fields to Business object attributes. The screen shot below illustrates this:

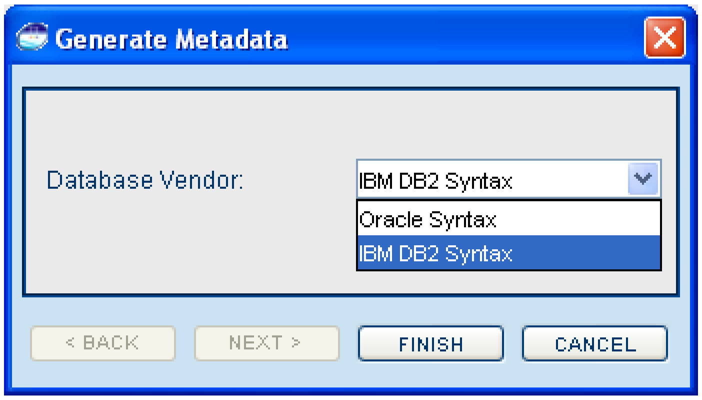

Financial Process Integrator
Financial Process Integrator
Workspace
Edit
Window
Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Host Systems
System: Sahara
SubSystem: sahara-accs
Transaction: txn001a
Request
Response
Duplicate Row
Remove Row
Clear
Expand
Contract
Add
Financial Process Integrator

name	Occurs	FinancialObjectPackage	FinancialObjectName	FinancialObjectAttribute	DefaultValue
Card_Number	[1-1] Expand	com.bankframe.bo.sha...	RoleExclusion	MUTUALLY_EXCLUDE...	1234567890
Account_Number	[1-2]	com.bankframe.bo.sha...	Actor	ACTOR_ID	
Account_Name	[1-2]	com.bankframe.bo.sha...	AccessChannel	CHANNEL_ID	Kevin Smith
Street_Address	[1-2] E	ankframe.bo.sha...	Role	PARENT_ROLE_ID	
State	[1-2]	ankframe.bo.sha...	RoleExclusion	MUTUALLY_EXCLUDE...	
Postcode	[1-2]	ankframe.bo.sha...	Role	MAX_NUMBER_OF_US...	
Account_Number_Fou...	[1-1]	ankframe.bo.sha...	RoleExclusion	ACCESS_PROVIDER_ID	
Account_Number_Fou...	[1-1]	ankframe.bo.sha...	Actor	LAST_LOGGED_IN_DA...	
Account_Number_Eigh...	[1-1]	ankframe.bo.sha...	ActorActorGroup	ACTOR_GROUP_ID	

Currently Selected Workspace is: D:\eontec\FPI\resources\sample\workspace\XMLSample.fpi

15.7 Generating Metadata

Click on the Workspace menu item in the application window, and select the ‘Generate SQL’ menu item. This will bring up a wizard that prompts you to choose a metadata format, and a database vendor to generate the metadata.

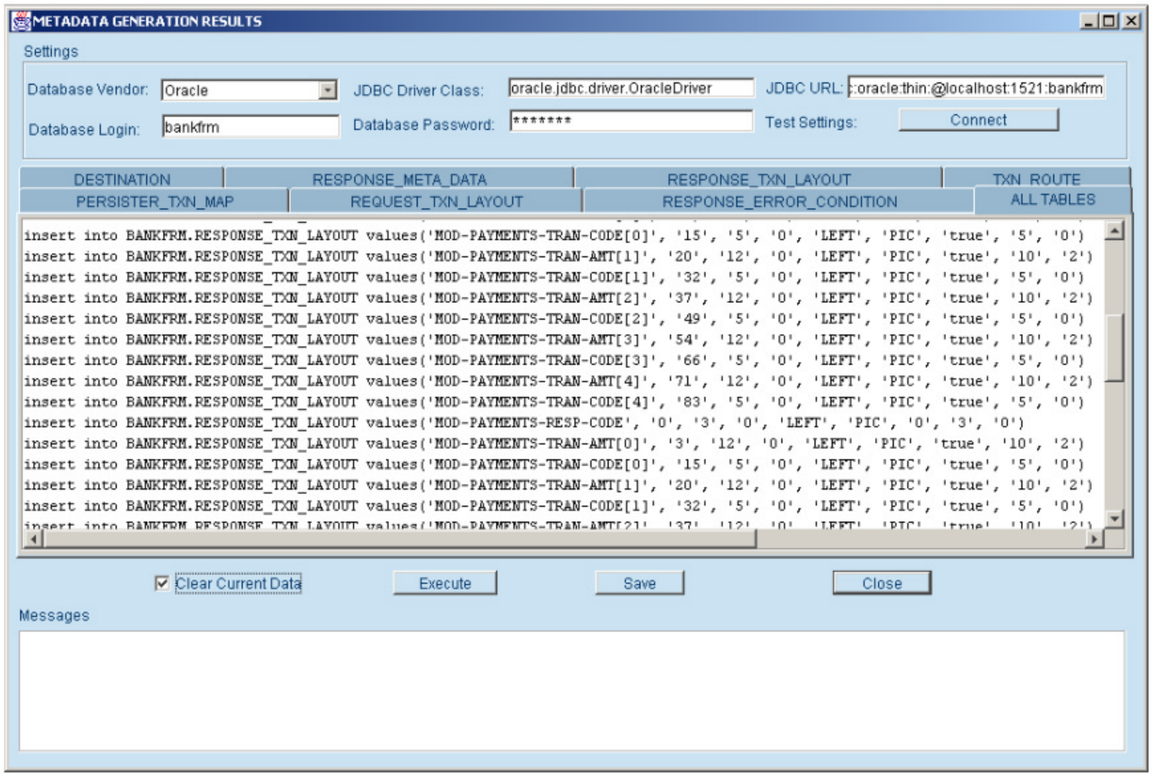


Database Vendor: IBM DB2 Syntax

Oracle Syntax
IBM DB2 Syntax

The Metadata schema formats are dependent on the database vendor option selected.

This will generate SQL insert statements based on the transactions, mappings and values you have defined within the Financial Process Integrator. It will bring up a panel displaying individual table values required for the Metadata format you selected. Each tab in the new window will have the required SQL Insert statements for various metadata tables, plus one tab representing all tables. You should see the resultant SQL in a pop up window like the following:



Database Vendor: Oracle
JDBC Driver Class: oracle.jdbc.driver.OracleDriver
JDBC URL: :oracle:thin:@localhost:1521:bankfrm
Database Login: bankfrm
Database Password: *******
Test Settings:

```
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[0]', '15', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[1]', '20', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[1]', '32', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[2]', '37', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[2]', '49', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[3]', '54', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[3]', '66', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[4]', '71', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[4]', '83', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-RESP-CODE', '0', '3', '0', 'LEFT', 'PIC', '0', '3', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[0]', '3', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[0]', '15', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[1]', '20', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-CODE[1]', '32', '5', '0', 'LEFT', 'PIC', 'true', '5', '0')
insert into BANKFRM.RESPONSE_TXN_LAYOUT values('MOD-PAYMENTS-TRAN-AMT[2]', '37', '12', '0', 'LEFT', 'PIC', 'true', '10', '2')
```

[✓] Clear Current Data

The section at the top of the window can be used if you wish to connect to a relational database system and execute your SQL statements from the Integrator. Once you have filled in the proper settings for the database system you wish to connect to, pressing the Connect button will test to see if a connection can be made. If the connection test is successful then you can press the Execute button, which is below the insert statements, to update the database immediately. By checking the Clear Current Data checkbox this ensures that any existing data will first be deleted from your database. Only the six tables DESTINATION, RESPONSE_META_DATA, RESPONSE_TXN_LAYOUT, TXN_ROUTE, PERSISTER_TXN_MAP, REQUEST_TXN_LAYOUT and RESPONSE_ERROR_CONDITION will be cleared of data. Any error messages or status updates from the attempt to execute will appear in the Messages text area at the bottom of the window. In addition, you also have the option to save the SQL file to your hard disk for later insertion and execution into your database tables.

16 Appendix A: FPI Configuration

This section describes Financial Process Integrator XML attributes and database settings which are configured in the BankframeConstants.properties file

Key	Sample Value	Description
HOST_MIDDLEWARE_TYPES	String[cobol,webmethods]	This sets the Host Middleware types that the Generic XML format supports
COMMON_FIELD_ATTRIBUTES	String[fieldName=REQUIRED]	This sets the Field attributes which are common across all Host Middleware Types
COBOL_FIELD_ATTRIBUTES	String[fieldOccurrences=REQUIRED,length=REQUIRED]	This sets the Field Attributes which are valid for COBOL only. To add additional middleware type Field attributes the Key will take the following format <Middleware Type>_FIELD_ATTRIBUTES **NOTE**: All Field Attributes must be written in the format <attributeName>=REQUIRED or <attributeName>=IMPLIED
COBOL_GROUP_ATTRIBUTES	String[groupName=REQUIRED, groupOccurrences=REQUIRED]	This sets the Group Attributes which are valid for COBOL only. To add additional middleware type Group attributes the Key will take the following format <Middleware Type>_GROUP_ATTRIBUTES **NOTE**: All Group Attributes must be written in the format <attributeName>=REQUIRED or <attributeName>=IMPLIED
SCHEMA_NAME_TEXT	BANKFRM	This sets the name of the Database Schema, which will be outputted in all SQL statements. e.g. insert into **BANKFRM**.REQUEST_TXN_LAY

Key	Sample Value	Description
		OUT values('txn001a', 'retacc01', 'Card_Number[0]', 1)
DB2_URL_TEXT	jdbc:DB2:FPI	This sets the DB2 URL and will appear in the JDBC URL: textfield on the Metadata generation results dialog
DB2_DRIVER_TEXT	COM.ibm.db2.jdbc.app.DB2Driver	This sets the DB2 JDBC driver class and will appear in the JDBC Driver Class: textfield on the Metadata generation results dialog
ORACLE_URL_TEXT	jdbc:oracle:thin:@database:1521:orcl	This sets the Oracle database URL and will appear in the JDBC URL: textfield on the Metadata generation results dialog
ORACLE_DRIVER_TEXT	oracle.jdbc.driver.OracleDriver	This sets the Oracle JDBC driver class and will appear in the JDBC Driver Class: textfield on the Metadata generation results dialog
FPI_DB_TABLE_NAMES	String[DESTINATION,RESPONSE_META_DATA]	This sets the names of the tables which will be cleared when the user checks the clear current data checkbox and clicks on the Execute button.

17 Appendix B: Cobol Copybooks

A standard Cobol program contains four divisions:

- Identification Division
- Environment Division,
- Data Division,
- Procedure Division.

A Cobol copybook is a data file which contains only Cobol data division. In the following, a simple description of the cobol copybook language rules will be given. All other syntaxes, if they are not in the following list, are not implemented in the current version of the parser.

17.1.1 Character Set

The characters A through Z, a through z, and blank are alphabetic; 0 through 9 are numeric; and + - * / = $, . " ( ) < > : are special. The upper-case and lower-case characters are equivalent, except when they are in a character string. The quotation (") delimits character strings. VS Cobol II (*) allows nonnumeric literals to be a maximum of 160 characters long.

17.1.2 Statement Format

The Cobol statement must be coded in certain columns for historical reasons. The constraints on the columns are:

- 1 to 6 contains an optional sequence number.
- 7 indicates the continuation of literals. Also used for comments debugging statements and page ejects (form feeds).
- 8 to 12 is itemed the A-area. Procedure names begin in this area. Column 8 is termed the A margin.
- 12 through 72 are termed the B-area. Statements begin in this area. Column 12 is termed B margin.
- 73 through 80 are available for program identification. These often contain sequence numbers. They are optional in Cobol copybooks.

The sequence number sections 1) and 5) are ignored when the cobol copybooks pass through the parser. The continuation of literals 2) only uses in the parser.

A statement in a cobol copybook can take one or more lines with a period as terminator at the end. If a statement exceeds one line, continue the statement on the next line in column 12 or later. There are continuation indicators in column 7. If the continuation indicator at column 7 is a space ( ), then the last character of the line being continued is assumed to be followed by a space. If the continuation indicator at column 7 is a hyphen (-), the continuation follows the last nonblank character of the first line.

In this current implementation of the parser, all other continuation indicators, such as (*), (D) and (/), are assumed as the line only contains comments in both A-area and B-area. All comments are ignored in the parser.

17.1.3 Cobol Words

A Cobol word consists of 1 to 30 characters. The characters may be 0 to 9, A to Z, a to z, or the hyphen (-). The hyphen must not be in the first or last character, but all other characters may appear in any position. There are about over 500 Cobol reserved words. We will only discuss a few of them which are related to Cobol copybooks in later sections.

17.1.4 Data Descriptions

Cobol copybooks contain data with a tree structure. The data can include scalars and vectors. The data as a whole is called a record. Items within a record that are not further subdivided are elementary items, which are leaves in the data tree. The other items are called group items.

17.1.5 Elementary Data Items

The common syntax for elementary items is:

```
<nn> <data-name> PIC  <character-string> [usage-clause] [VALUE <value>]
```

`nn`	`level-number`. A qualified `level-number` is a one or two digits number, which can be 01 to 49(or 1 to 49), 66, 77, and 88. Level-numbers 66, 77 are not implemented in the current parser.
`data-name`	Cobol word. Omit the name or code FILLER as the data name to indicate an unnamed item. PIC, PICTURE or PICTURE IS - clause. These are reserved words. It specifies the form of the followed character-string.
`character-string`	There are two types of the character-strings: One is number, and the other is character. We'll look at these in another section.
`usage-clause`	works only with numeric character-string. It contains the way for computers to store the data. The supported usage-clause values in the parser are: • "by default" means there are no more clauses between alphanumeric characters and VALUE clause • COMP, USAGE COMP, or USAGE IS COMP • COMP-3, USAGE COMP-3, or USAGE IS COMP-3
`value`	It marks that the variable will have the initial value, which is given by the value after the reserved word `VALUE`.

17.1.6 Group Items

The common syntax for group items is:

```
<nn> <group-name>       [OCCURS <n> TIMES]

<nn> <group-name>       OCCURS <n1> To <n2> DEPENDING ON <item-name>
```

Where the `OCCURS` clause is optional. Both `OCCURS` clauses describe fixed-length tables. The first gives the table size in `n`, and the second gives the domain of this number but stores the number of group instances in a data item called `item-name`.

17.1.7 Character String Types

There are two types of `character-string`; one is number, and the other is character.

17.1.7.1 Alphanumeric Character Types

The alphanumeric character has only the following form:

- `X(<n>)`: `X` stands for alphanumeric character, which can consist of any of the characters in Cobol character set, including alphabetic and numeric characters. It can be also represented by for example `XXXXXX`, which is the same as `X(6)`. The alphabetic data items are the same as alphanumeric items, except a different form: `A(<n>)`. The maximum number of alphanumeric characters in a character is 249 in VS COBOL II.

17.1.7.2 Numeric Character Types

The number has the following forms:

- `9(<n>)V9(<m>)`: `n` unsigned digits before the decimal point, and `m` unsigned digits after the decimal point, where `V` specifies the assumed decimal point. The decimal point does not count in determining the length.
- `S9(<n>)V9(<m>)`: `n` signed digits before the decimal point, and `m` unsigned digits after the decimal point, where `V` specifies the assumed decimal point. The decimal point does not count in determining the length. `S` does not count in determining the length.
- `9(<n>)`: `n` unsigned digits. It assumes the decimal point is on the right side. That is `9(n)` is the same as `9(n)V`. It can be also expressed by for example `9999`, which is the same as `9(4)`.
- `S9(<n>)`: `n` signed digits. It also can be expressed by for example `S99999`, which is the same as `S9(5)`. Similarly there are `V9(<n>)` and `SV9(<n>)`. The maximum number of digits that can be contained in a numeric data item is 18, that is m, n<= 18.

17.1.8 Conditions

The `level-number` 88 has a special meaning. The item with `level-number` 88 is called `condition-name`. It specifies some of the logic in the data descriptions. The common syntax for a `condition-name` is

```
88 <condition-name> VALUE <n>
```

where `<n>` can be a number or a string. For example,

```
10 FIELD0 PIC X.
      88 COND0                                              VALUE "X".
      88 COND1                                              VALUE "Y".
      88 COND2                                              VALUE "Z".
```

In the above the field '`FIELD0`' can take the values '`X`', '`Y`' or '`Z`'.

Reference: G. DeWard Brown, Advanced ANSI COBOL with Structured Programming, 2nd edition, Wiley Professional Computing, Wiley & Sons, Inc, 1992.

18 Appendix C: Recurring Records

As we have seen in the appendix on copybooks, a field can occur many times within an instantiation of a copybook. We call these possible occurrences. It's improbable but possible for a possible occurrence to map to one bean field and another occurrence of the same field to map to another bean field. Note that these are **possible** occurrences signifying that they may not exist in any instantiation of a copybook. This can happen with the '`OCCURS FROM 1..10 DEPENDING ON COUNT-FIELD`' syntax. In this case it's not known what mappings will be used until runtime.

19 Appendix D: FPI Meta Data Rules

19.1 Cobol Copybook Fields

property	Description	Populated	Editable	Valid Input	Used In
(Group) Level	Level (depth) of the current cobol group	On import of cobol	Yes	Numeric as String	not used in metadata
Name	Transaction Field Name	On import of cobol	No	Alphanumeric as String	request_txn_layout, response_txn_layout
Value	Field Default Value	By user on Txn Records tab	Yes	Alphanumeric as String	request_txn_layout
Length	Field Length	On import of cobol	No	Alphanumeric as String	request_txn_layout, response_txn_layout
Level	Level in Record	On import of cobol	No	Alphanumeric as String	not used in metadata
DataType	Field Data Type	On import of cobol	No	Alphanumeric as String	request_txn_layout, response_txn_layout (null entries default to 'ASCII' type)
*IsFiller	Is Field Filler?	On import of cobol	No	Boolean as String (true, false)	not used in metadata
*IsAlphabetic	Is Field Alphabetic?	On import of cobol	No	Boolean as String (true, false)	not used in metadata
IsNumeric	Is Field Numeric?	On import of cobol	No	Boolean as String (true, false)	not used in metadata
*IsSigned	Is Field Signed?	On import of cobol	No	Boolean as String (true, false)	request_txn_layout, response_txn_layout
NumBefore	Number of	On import of	No	Integer as String	request_txn_layou

property	Description	Populated	Editable	Valid Input	Used In
	spaces before decimal (Numeric data only)	cobol			t, response_txn_layout
*NumAfter	Number of spaces after decimal (Numeric data only)	On import of cobol	No	Integer as String	request_txn_layout, response_txn_layout
*Allignment	Field Allignment	On import of cobol	No	String (RIGHT, LEFT)	request_txn_layout, response_txn_layout
*FillCharacter	Fill Character	On import of cobol	No	String ('0' for Numeric fields & ' ' for Character fields)	request_txn_layout, response_txn_layout
ErrorId	Error Id (For fields that identify error conditions – Set in your response mapping table)	By user after record import	Yes	Alphanumeric as String	
ErrorCondition	Condition identifying an error (Error Fields Only)	By user from drop-down list after record import	Yes	String chosen from Set List	response_error_condition
ErrorValue	Value that satisfies error condition above (Error fields only)	By user after record import	Yes	Alphanumeric as String	response_error_condition
ErrorCombineNext	Combine this error condition with the next one? (Error fields only)	By user after record import	Yes	Boolean as String (yes, no)	response_error_condition

property	Description	Populated	Editable	Valid Input	Used In
ErrorSequence	Sequence (order) of this error condition (multiple error fields only)	By user after record import	Yes	Integer as String	response_error_condition
ErrorCode	Error Code (Error Fields only)	By user after record import	Yes	Alphanumeric as String	response_error_condition
ErrorType	Error Type (Error Fields only)	By user after record import	Yes	Alphanumeric as String	response_error_condition

19.2 Component Fields

None of the Component Fields are editable. If these need to be changed, it should be done in the object model within Rational Rose

19.2.1 Host System Fields

property	Description	Populated	Editable	Valid Input	Used In
Name	Logical Name of Host System	By user on creation of system	Yes	Alphanumeric as String	not used in metadata
Description	Description of Host System	By user on creation of system	Yes	Alphanumeric as String	not used in metadata
Vendor	Vendor of Host System	By user on creation of system	Yes	Alphanumeric as String	not used in metadata

19.2.2 Connector Fields

property	Description	Populated	Editable	Valid Input	Used In
DataFormatter	Full package and name of the DataFormatter class for this connector	By user after import of Connectors	Yes	Alphanumeric as String	txn_route
Properties	Configuration	By user after	Yes	Alphanumeric as	destination

property	Description	Populated	Editable	Valid Input	Used In
	properties for this connector	import of Connectors		String (key-value property pairs are semi-colon delimited)	

19.2.3 SubSystem Fields

Property	Description	Populated	Editable	Valid Input	Used In
Name	Logical name of the subsystem	By user on creation of subsystem	Yes	Alphanumeric as String	not used in metadata
Description	Description of the subsystem	By user on creation of subsystem	Yes	Alphanumeric as String	not used in metadata
HostSystem	Name of the host system that this subsystem falls under	By user on creation of subsystem from drop-down list.	Yes	String chosen from set list	not used in metadata

19.2.4 Host Transaction Fields

Property	Description	Populated	Editable	Valid Input	Used In
Transaction Code	Code for this transaction	By user on creation of transaction	Yes	Alphanumeric as String	request_txn_layout, response_meta_data, response_error_condition, persister_txn_map, txn_route
Transaction Type	Type of this transaction	By user on creation of transaction	Yes	Alphanumeric as String	request_txn_layout, response_meta_data, response_error_condition, persister_txn_map, txn_route

Property	Description	Populated	Editable	Valid Input	Used In
Description	Description of this transaction	By user on creation of Transaction	Yes	Alphanumeric as String	not used in metadata
EnglishName	Explanation (in plain English) of the transaction code & type	By user on creation of Transaction	Yes	Alphanumeric as String	not used in metadata
HostSystemName	Name of the host system that this transaction falls under	By user on creation of txn from drop-down list	Yes	String chosen from set list	not used in metadata
SubSystem Name	Name of the subsystem that this transaction falls under	By user on creation of txn from drop-down list	Yes	String chosen from set list	Not used in metadata
HostConnector	Name of the connector that this transaction uses	By user on creation of txn from drop-down list	Yes	String chosen from set list	destination

19.2.5 Persister Fields

Property	Description	Populated	Editable	Valid Input	Used In
EntityName	Full package and class name of Entity bean	On import of model	No	String chosen from set list	persister_txn_map
EntityMethodName	Method name from the entity selected	On import of model	No	String chosen from set list	persister_txn_map
HostConnectorName	Cache Policy	Drop Down from list of imported connectors	Yes	String chosen from set list	persister_txn_map (Cache_Policy)

19.2.6 Request Mapping Fields

Property	Description	Populated	Editable	Valid Input	Used In
Txn Field	Name of the host record field for this transaction	From selected Transaction Record	No	Alphanumeric as String	request_txn_layout
Occurs	Indicates whether or not this field occurs multiple times	Parsed from imported transaction record	No	[n-n], where n is Numeric as String	Indirectly used to create repeating entries in the request_txn_layout table
Object Package	Full package name of entity bean used in mapping	As drop-down list from imported model	Yes	String chosen from list	No used in metadata
Object Name	Java class name of entity bean used in mapping	As drop-down list from imported model	Yes	String chosen from list	Not used in metadata
Object Attribute	Java attribute name of entity bean used in mapping	As drop-down list from imported model	Yes	String chosen from list	request_txn_layout

19.2.7 Response Mapping Fields

Property	Description	Populated	Editable	Valid Input	Used In
Txn Field	Name of the host record field for this transaction	From selected Transaction Record	No	Alphanumeric as String	request_txn_layout
Object Package	Full package name of entity bean used in mapping	As drop-down list from imported model	Yes	String chosen from list	not used in metadata

Property	Description	Populated	Editable	Valid Input	Used In
Object Package	Full package name of entity bean used in mapping	As drop-down list from imported model	Yes	String chosen from list	not used in metadata
Object Name	Java class name of entity bean used in mapping	As drop-down list from imported model	Yes	String chosen from list	response_meta_data
Object Attribute	Java attribute name of entity bean used in mapping	As drop-down list from imported model	Yes	String chosen from list	response_meta_data
Error Field	Indicates whether this field is used to flag errors from the host	By user as checkbox	Yes	Boolean parsed from checkbox	response_error_condition