Financial Process Integrator Guide

Version 2005

May 2005

Contents

1 What's New in this Release

The following changes have been introduced in the Financial Process Integrator, version 2005:

Topic	Description
The Cobol Copybook Example chapter has been removed.	The Financial Process Integrator no longer supports the importing of Cobol Copybooks in raw format. All data types must now imported in XML format via a common interface. The XML transaction definition must adhere to the DTD format of FPITransaction.dtd. FPITransaction.dtd is described on page 28
Color Coding of Transaction Fields, page 26	Color coding has been added to Transaction Field mapping for usability.
Handling Multiple Responses, page 26	The ability for multiple Response Records to be defined for a Host Transaction has been added.
Mapping Recurring Transaction Response Fields, page 29	Recurring fields are now supported within groups or recurring groups. Groups can be nested if the `occurrences` value is set to 1 for all the nested groups. Nested occurring groups are not supported, that is, recurring groups are not supported within recurring groups.
FPITransaction DTD, page 33	The following changes have been made to FPITransaction.dtd: The TransactionOverview Group element attribute groupName CDATA #REQUIRED has been changed to name CDATA #REQUIRED. The GroupOverview Field element attribute fieldName CDATA #REQUIRED has been changed to name CDATA #REQUIRED.
Amending Transaction Records, Groups and Fields, page 37	The following can now be amended: a Transaction Record that has previously been imported, the attributes of a Transaction Field and the attributes of a Transaction Group.
Re-importing an SRF Model, page 46	The FPI tool now allows a Model to change and be re-imported without having to re-map information. If you have mapped an attribute belonging to a Financial Object to a host transaction field and this attribute has been renamed in the model you are importing, the mapping information is automatically updated to reflect this change.
Broker Request and Response	Broker Request and Response Objects have been

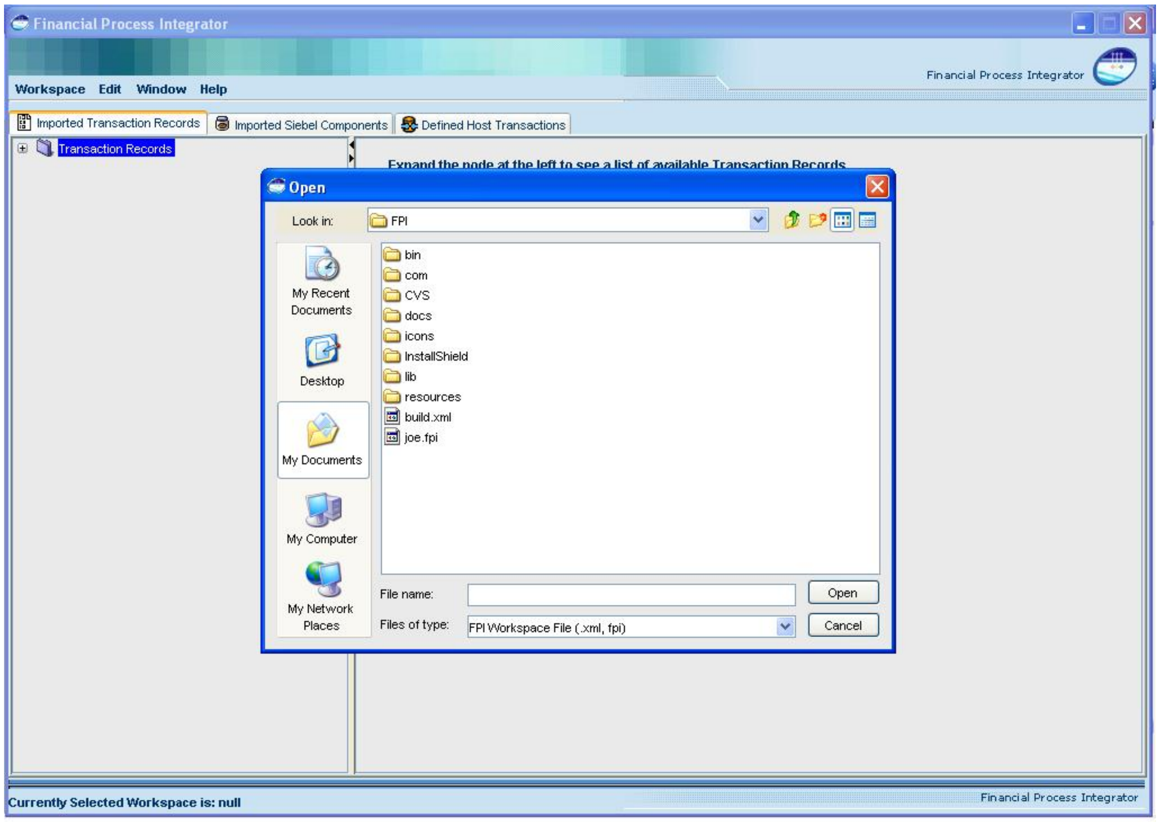
Topic	Description
Objects, page 49	introduced to define parameters that do not map to the attributes in the Financial Objects. Broker Request and Response Objects are used in conjunction with the SessionAmendHelper and the TransactionHandlerBroker.
Refreshing MetaData, page 59	A Refresh MetaData utility has been added to the FPI tool. If new user-defined columns are added to the RESPONSE_META_DATA table the Refresh Meta Data utility identifies these columns, updates the BankFrameConstants.properties file and the MetaData mappings in the applicable Response nodes of the corresponding Host Transaction on the Defined Host Transactions tab. When the MetaData SQL is then regenerated the new user-defined columns are included in the SQL.
Cobol Copybook Conversion, page 61	Transaction Records must be in XML format, compliant with the FPITransaction DTD, before being imported into the Financial Process Integrator tool. The Cobol Copybook Conversion utility is now provided to help convert a Cobol Copybook to FPITransaction.dtd format.
Cache Indexing Support	The INDEX_NAME column has been added to the PERSISTER_TXN_MAP table to support cache indexing. Refer to the MCA Services Developer Guide for more information on Cache Indexing.

2 Getting Started

The Financial Process Integrator tool can be used to import, define and manipulate software components that are required to map Siebel Retail Finance 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 MetaData that is required for the MCA Services Financial Process Integrator engine to execute these transactions. The Metadata maps a host transaction, in the form of a set of host transaction records, to a set of financial objects.

This document assumes familiarity with Financial Components and the MCA Services Financial Process Integrator engine; both are covered in the Financial Process Integration chapter of the MCA Services Developer Guide.

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



Expand the node at the left to see a list of available Transaction Records

Currently Selected Workspace is: null

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.

About the Financial Process Integrator 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.

Working in the FPI Workspace

The following operations are available on the FPI Workspace file menu:

- Creating a New Workspace File
- Opening a Workspace File
- Saving a Workspace File
- Generating SQL
- Testing Transactions

Creating a New Workspace File

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

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.

Saving a Workspace File

The Save/Save AS menu items allow a user to save all data that is currently in the Host Tool's 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.

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.

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.

3 Host Systems

The Host Systems list node 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.

Host System Operations

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

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.

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 on 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:

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.

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.

Importing Host Connectors

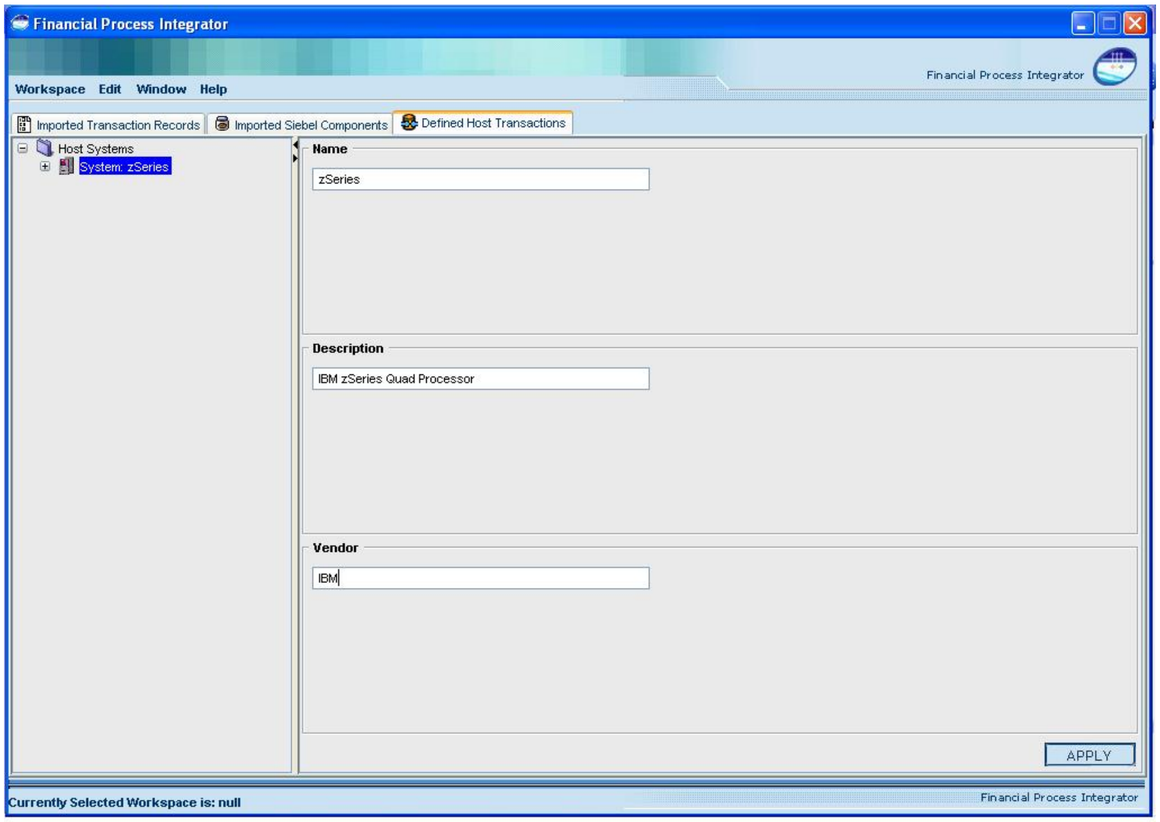
The Import Connectors menu item will launch a wizard that allows you to import Host Connectors into the FPI workspace. Host Connectors are defined in the main configuration file used by MCA Services. This file is named `BankframeResource.properties`, and is usually located in the directory of the application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

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

4 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.

Sub System Operations

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

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.

Removing a Host Sub System

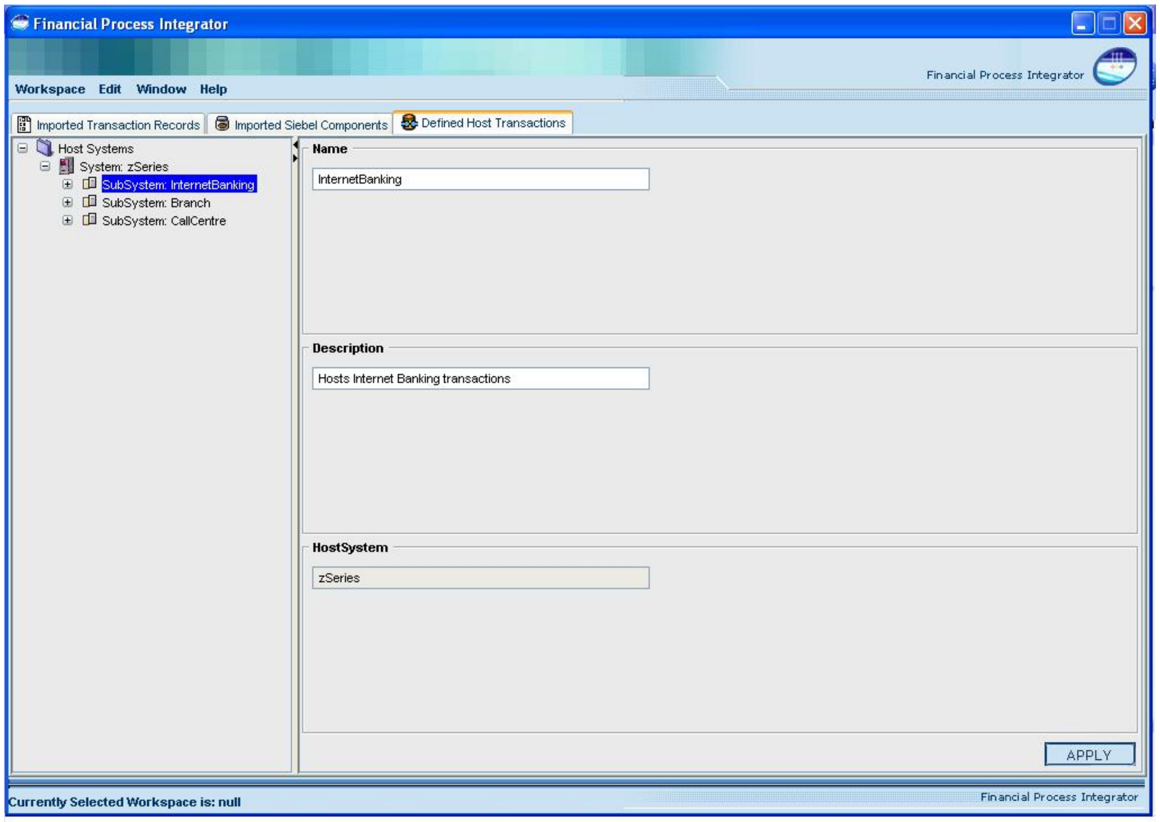
The Remove menu item will remove the host sub system that is currently selected from the workspace. When a host sub system is removed, it will also remove all child node objects that are below it, therefore any Host Transactions and Mappings that have been added to this Host Sub System will also be removed.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

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
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

5 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 financial 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 tool to generate the SQL statements required to populate the metadata used by the MCA Services Financial Process Integrator.

Host Transaction Operations

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

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.

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 Developer Guide.

Testing a Host Transaction

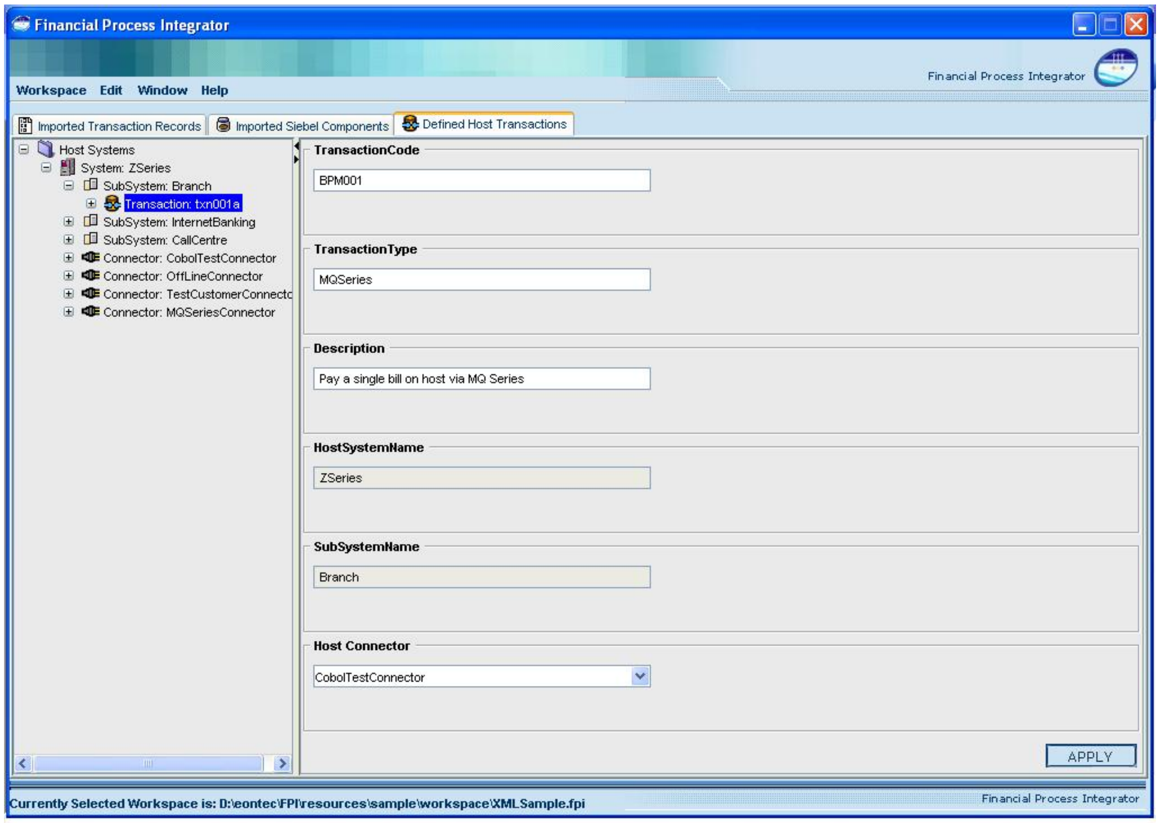
The Test Transaction menu item will allow a user to test host transactions that have been defined in the Financial Process Integrator. Testing a transaction will create and send a HTTP request to a deployed instance of MCA Services and display the results.

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: TestCustomerConnecto
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

6 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 Services Financial Process Integrator. 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.

Persister Operation

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

Removing a Persister

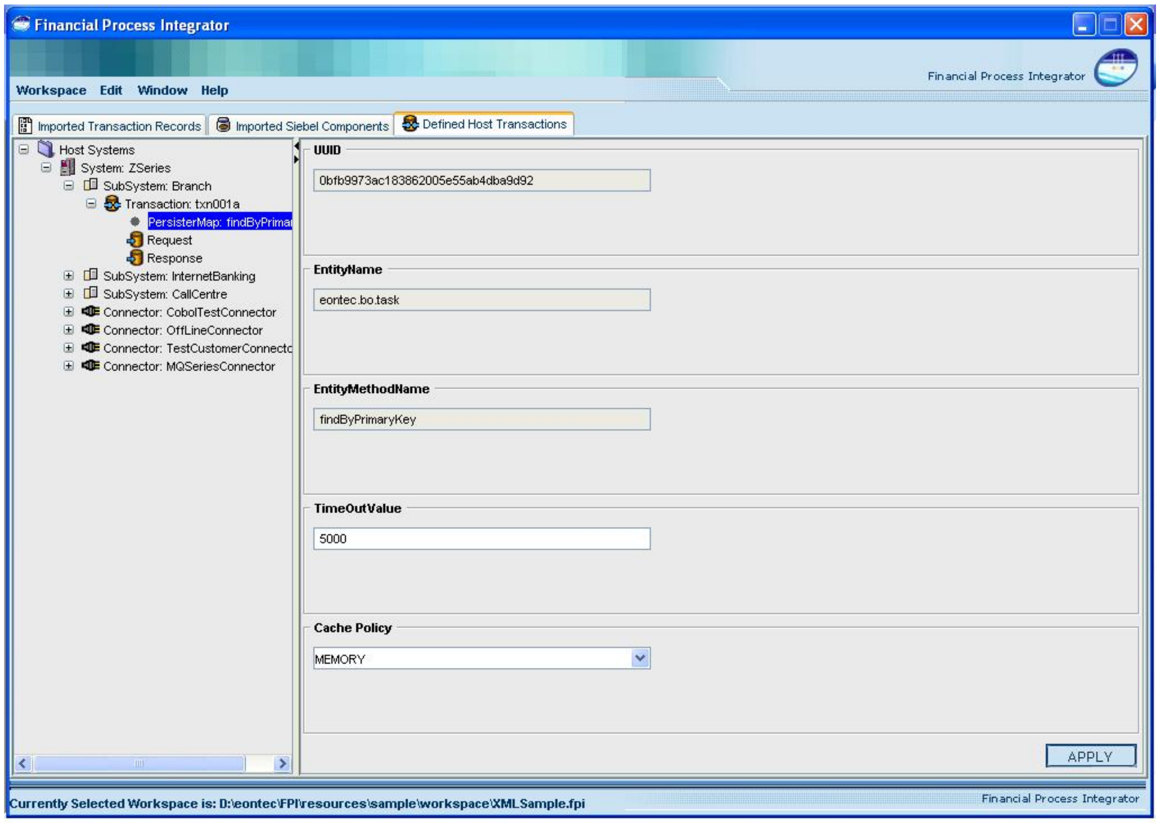
The Remove menu item will remove the Persister that is currently selected from the workspace.

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

UUID: 0bfb9973ac183862005e55ab4dba9d92

EntityName: eontec.bo.task

EntityMethodName: findByPrimaryKey

TimeOutValue: 5000

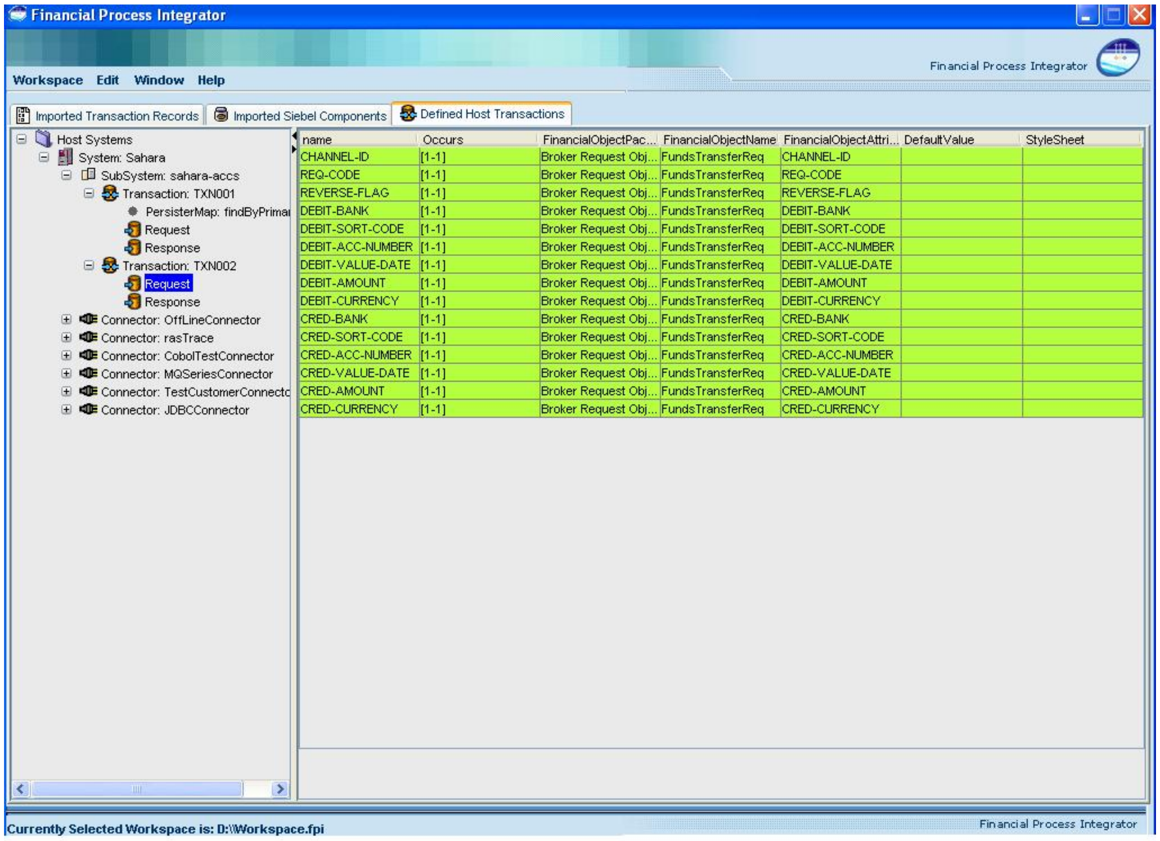
Cache Policy: MEMORY

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

7 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 financial 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.


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: TXN001
PersisterMap: findByPrimai
Request
Response
Transaction: TXN002
Request
Response
Connector: OffLineConnector
Connector: rasTrace
Connector: CobolTestConnector
Connector: MQSeriesConnector
Connector: TestCustomerConnectc
Connector: JDBCConnector
Currently Selected Workspace is: D:\\Workspace.fpi
Financial Process Integrator

name	Occurs	FinancialObjectPac...	FinancialObjectName	FinancialObjectAttri...	DefaultValue	StyleSheet
CHANNEL-ID	[1-1]	Broker Request Obj...	FundsTransferReq	CHANNEL-ID		
REQ-CODE	[1-1]	Broker Request Obj...	FundsTransferReq	REQ-CODE		
REVERSE-FLAG	[1-1]	Broker Request Obj...	FundsTransferReq	REVERSE-FLAG		
DEBIT-BANK	[1-1]	Broker Request Obj...	FundsTransferReq	DEBIT-BANK		
DEBIT-SORT-CODE	[1-1]	Broker Request Obj...	FundsTransferReq	DEBIT-SORT-CODE		
DEBIT-ACC-NUMBER	[1-1]	Broker Request Obj...	FundsTransferReq	DEBIT-ACC-NUMBER		
DEBIT-VALUE-DATE	[1-1]	Broker Request Obj...	FundsTransferReq	DEBIT-VALUE-DATE		
DEBIT-AMOUNT	[1-1]	Broker Request Obj...	FundsTransferReq	DEBIT-AMOUNT		
DEBIT-CURRENCY	[1-1]	Broker Request Obj...	FundsTransferReq	DEBIT-CURRENCY		
CRED-BANK	[1-1]	Broker Request Obj...	FundsTransferReq	CRED-BANK		
CRED-SORT-CODE	[1-1]	Broker Request Obj...	FundsTransferReq	CRED-SORT-CODE		
CRED-ACC-NUMBER	[1-1]	Broker Request Obj...	FundsTransferReq	CRED-ACC-NUMBER		
CRED-VALUE-DATE	[1-1]	Broker Request Obj...	FundsTransferReq	CRED-VALUE-DATE		
CRED-AMOUNT	[1-1]	Broker Request Obj...	FundsTransferReq	CRED-AMOUNT		
CRED-CURRENCY	[1-1]	Broker Request Obj...	FundsTransferReq	CRED-CURRENCY		

8 Transaction Response

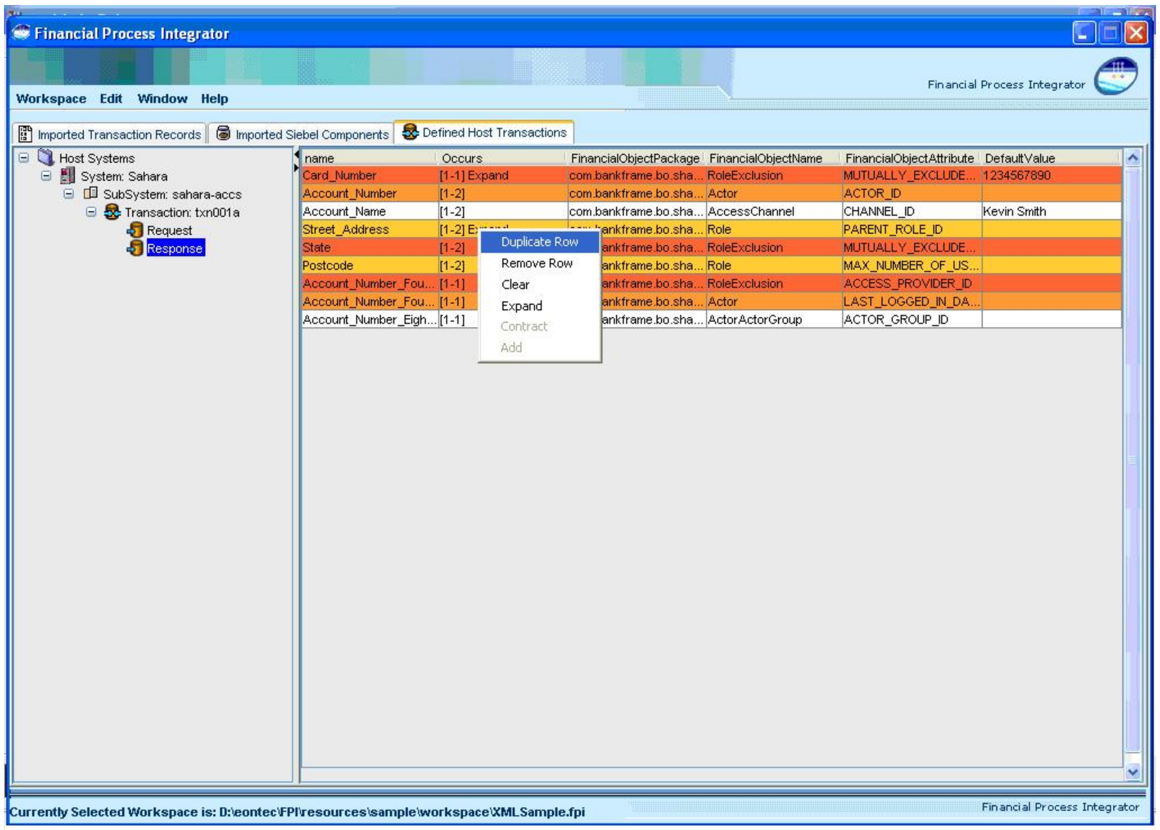
The Transaction Response Mapping Table

The mapping table of the Response node on the Defined Host Transactions tab is slightly different to 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 Transaction Response record to attributes of your financial objects.

For each Response Transaction Field (row) in this table, the user selects a package, financial object and financial object attribute from the drop-down menu that maps to the transaction record field. The table cells for these properties are drop-down lists that are populated from all of the financial 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

The divider bar between the left and right window panes can be moved to provide a larger view of the 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.

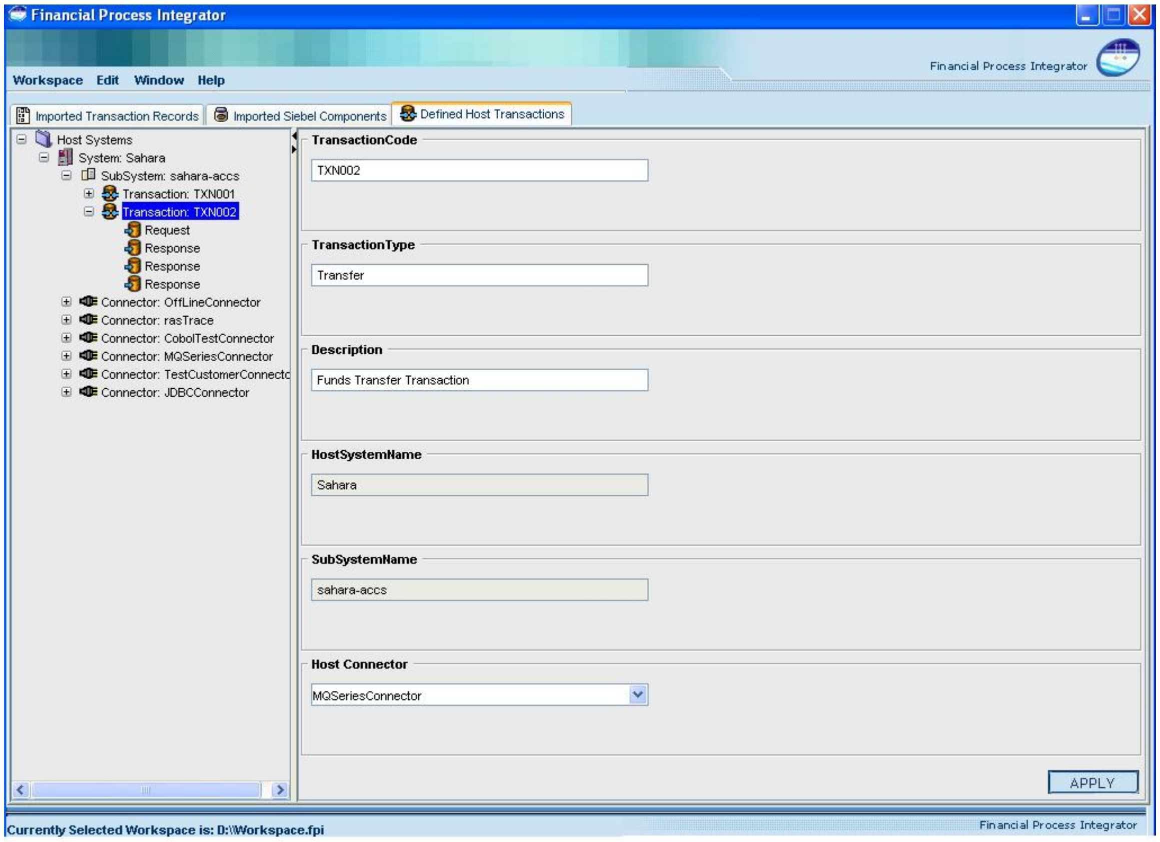
Color Coding of Transaction Fields

Color coding has been added to Transaction Field mapping for usability. The background color for all mapped fields is automatically set to a color other than white. The background color for any Fields that are not mapped remains white. If you map a Transaction Field(A) to a Financial Object Attribute(B) a color is applied to the corresponding row. If you map a second Transaction Field(C) to a Financial Object Attribute(D) where both Financial Object Attributes B and D belong to the same Financial Object, the same color is applied to the two rows.

Color coding of Transaction fields is turned on by default. If you want to switch off color coding of transaction fields set the MAPPING_BG_COLOR property in the Integrator.properties file to false.

Handling Multiple Responses

The Financial Process Integrator enables multiple Response Records to be defined for a Host Transaction. The screen shot below illustrates a Host Transaction that has more than one Response Record defined.


Financial Process Integrator
Workspace
Edit
Window
Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Host Systems
System: Sahara
SubSystem: sahara-accs
Transaction: TXN001
Transaction: TXN002
Request
Response
Response
Response
Connector: OffLineConnector
Connector: rasTrace
Connector: CobolTestConnector
Connector: MQSeriesConnector
Connector: TestCustomerConnecto
Connector: JDBCConnector
APPLY

TransactionCode: TXN002

TransactionType: Transfer

Description: Funds Transfer Transaction

HostSystemName: Sahara

SubSystemName: sahara-accs

Host Connector: MQSeriesConnector

Currently Selected Workspace is: D:\\Workspace.fpi

To remove a Response record, right-click on the record and select Remove.

Mapping Transaction Response Fields

Mapping a Transaction Field to a Financial Object

To map a Transaction Field to a Financial Object

1. Navigate to the Defined Host Transactions tab.
2. Select the Transaction Response in the left window.
3. Select the Transaction Response row in the right window that represents the response field to be mapped.
4. Select a FinancialObjectPackage, FinancialObjectName and FinancialObjectAttribute from the drop-down menus.

Mapping a Transaction Field to more than one Financial Object Attribute

This mapping allows a Transaction field to initialize the state of many different Financial object attributes.

To map a Transaction Field to more than one Financial object Attribute

1. Navigate to the Defined Host Transactions tab.
2. Select the Transaction Response in the left window.
3. Right-click on the Transaction Field in the right window. The Transaction Response operations menu then displays.
4. Select Duplicate Row. An additional mapping row for the field is then added to the mapping table.
5. Select the additional FinancialObjectAttribute that the field is to be mapped to from the drop-down menu.

Removing a Transaction Response Field from the Response Mapping Table

- Right-click on a Transaction Response field and select the Remove Row menu item.

Removing Response Transaction Field Mapping

- Right-click on a Transaction Response field and select the Clear menu item to remove the mapping information defined for the currently selected response transaction field.

Displaying all Occurrences of a Transaction Field

- Right-click on a Transaction Response field and select the Expand menu item to display all occurrences of the selected field.

Displaying One Instance of a Transaction Field

- Right-click on a Transaction Response field and select the Contract menu item to display one single occurrence of the selected field.

Restoring a Removed Response Transaction Field

- Right-click on a Transaction Response field and select the Add menu item to display and restore any response transaction fields that where previously removed.

Mapping Recurring Transaction Response Fields

Recurring fields are supported within groups and recurring groups. Groups can be nested if the `occurrences` value is set to 1 for all the nested groups. Nested occurring groups are not supported, that is, recurring groups are not supported within recurring groups.

The following Transaction Response operations available on the Defined Host Transactions tab are related to the handling of recurring fields:

Mapping Each Occurrence of a Recurring Transaction Field to a New Instance the Same Financial Object Attribute

If a Transaction Response field is defined such that it has a field occurrence value of n, and Expand is displayed for the Occurs value, each occurrence of the field can be mapped to a new instance of the same financial object attribute. If Contract is displayed for the Occurs value right-click and select Contract.

Mapping Each Occurrence of a Recurring Transaction Field to a Different Financial Object Attribute

To map each occurrence of the recurring field to a different financial object attribute, right-click on the recurring field, select Expand and map each occurrence to an attribute as required.

9 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.

Host Connector Operations

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

Removing a Host Connector

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

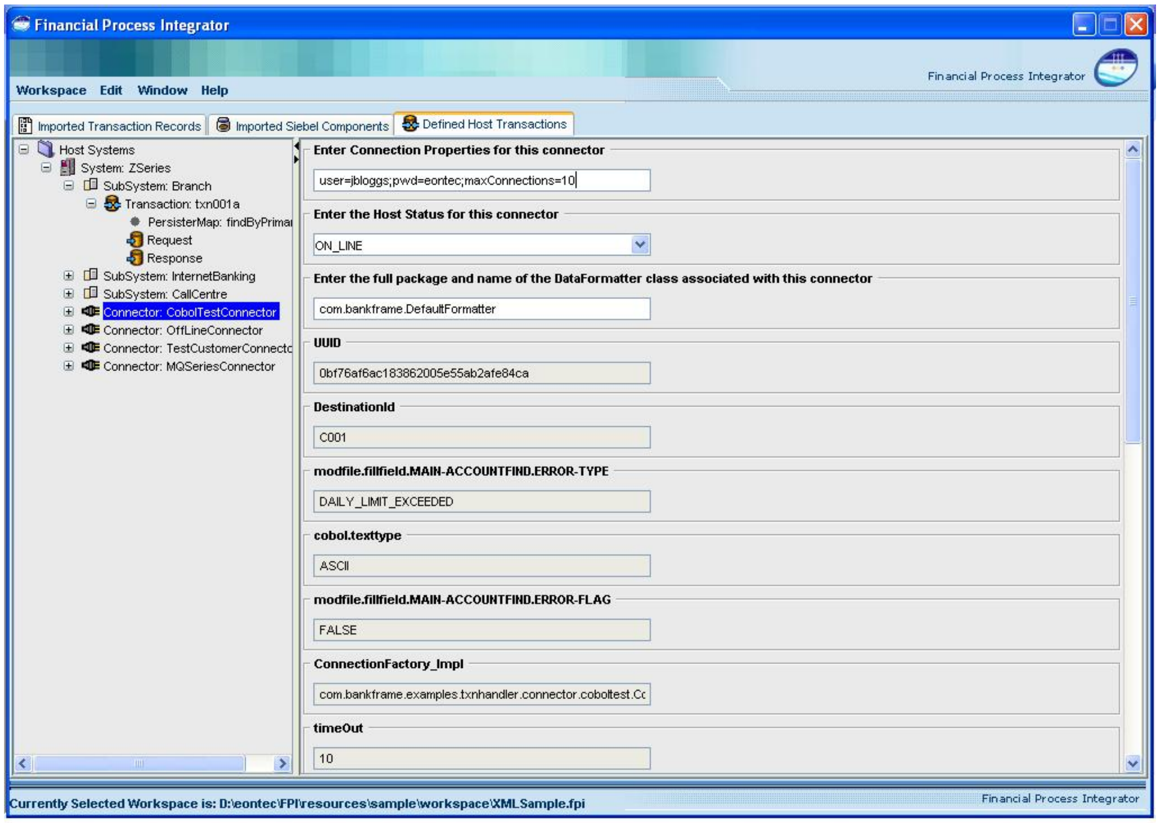
Host Connector Properties

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: findByPrimar
Request
Response
SubSystem: InternetBanking
SubSystem: CallCentre
Connector: CobolTestConnector
Connector: OffLineConnector
Connector: TestCustomerConnecto
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

10 Transaction Records

About 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. A Transaction Record represents a particular record or data type that is being used by your host system. The order of the Transaction Records is expressed via the XML transaction record definition.

About the FPITransaction DTD

All Transaction Record XML definitions 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

        name CDATA #REQUIRED

        groupOccurrences CDATA #REQUIRED

        redefines CDATA #IMPLIED

>

<!ELEMENT GroupOverview (#PCDATA)>

<!ELEMENT Field (FieldOverview?)>

<!ATTLIST Field

        fieldOccurrences CDATA #REQUIRED
        name 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]
```

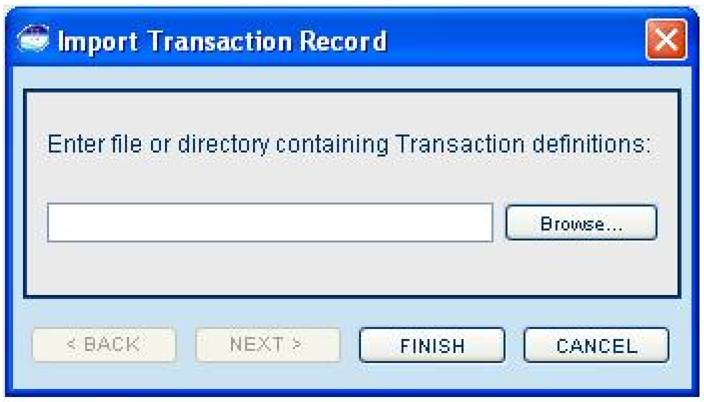
COMMON_FIELD_ATTRIBUTES represents the fields that are mandatory across all middleware types.

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

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
Enter file or directory containing Transaction definitions:
Browse...
< BACK
NEXT >
FINISH
CANCEL

The FPI automatically parses the record transaction type and stores its associated properties and structure in the workspace.

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.

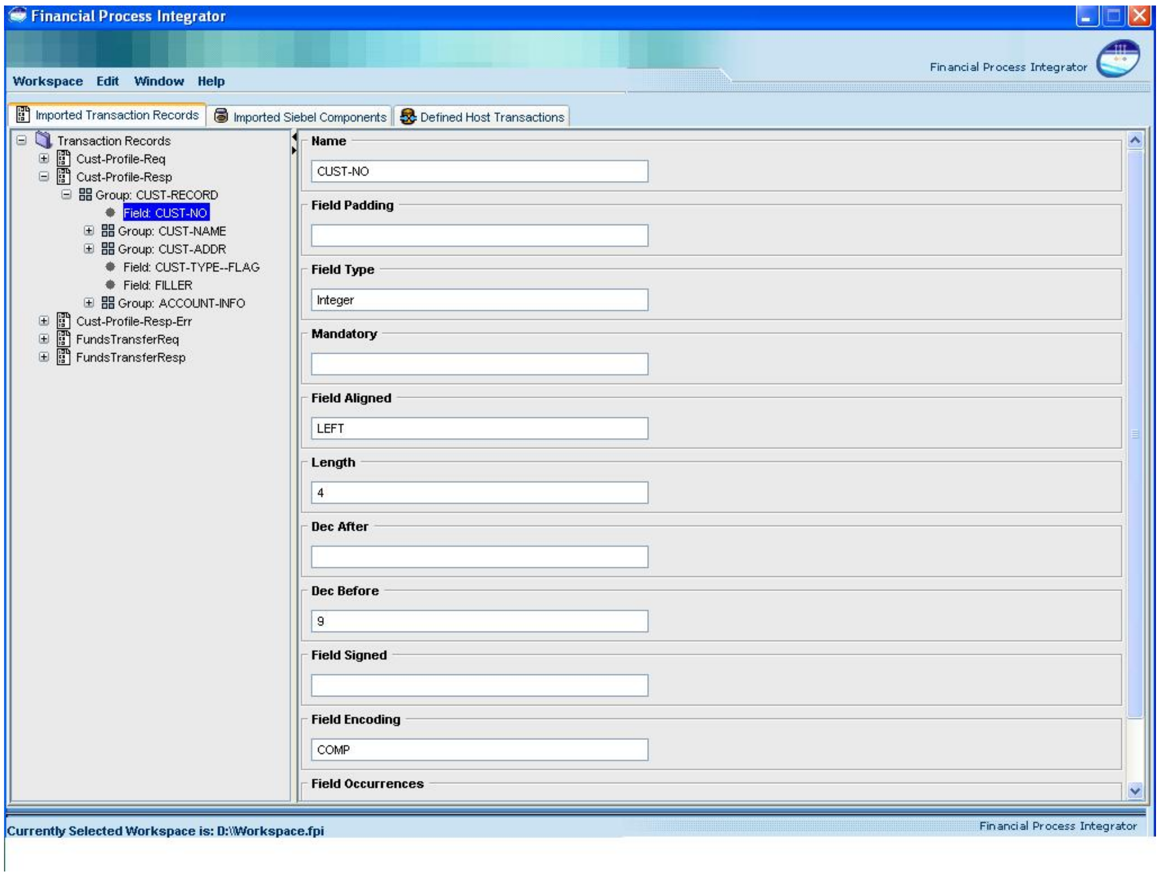
About Transaction Record Properties

The Transaction Record list node has no properties, but each Transaction Record as well as its sub groups, and fields has 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
name	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

XML Record (Middleware Type: Cobol Copybook)

The following screen capture 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
- Cust-Profile-Req
- Cust-Profile-Resp
  - Group: CUST-RECORD
    - Field: CUST-NO
    - Group: CUST-NAME
    - Group: CUST-ADDR
    - Field: CUST-TYPE--FLAG
    - Field: FILLER
    - Group: ACCOUNT-INFO
- Cust-Profile-Resp-Err
- FundsTransferReq
- FundsTransferResp

Name	CUST-NO
Field Padding	
Field Type	Integer
Mandatory	
Field Aligned	LEFT
Length	4
Dec After	
Dec Before	9
Field Signed	
Field Encoding	COMP
Field Occurrences	

Currently Selected Workspace is: D:\\Workspace.fpi

About Amending a Transaction Record, Group or Field

A Transaction Record that has previously been imported, the attributes of a Transaction Field and the attributes of a Transaction Group can be amended as follows:

- Existing attributes of Transaction Fields or Transaction Groups can be amended. These amendments can be made in the workspace on the right of the screen.
- New Transaction Fields or Transaction Groups can be added. These amendments to the structure of a Transaction Record can be made in the panel on the left of the screen. This functionality is drag and drop enabled: a user can drag a field or group and drop the dragged item(s) onto a group. The order of fields and groups within a Transaction Record is dictated by the order in the host transaction.

NOTE: The FPI does not allow a user to drag and drop items between Transaction Records. If the user amends Transaction Record A which is used by Host Transaction HT1, any changes made to A are automatically applied to HT1.

Amending a Transaction Record

The following actions can be performed on a Transaction Record:

- Insert a Child Field
- Insert a Child Group
- Remove a Child Group

NOTE: Ensure that any recurring field is contracted in the mapping screen before amending a Transaction Field or Group within the Transaction Records tab, otherwise in the mapping screen the occurs value for the first occurrence of this field will display as [n-n] Expand and should display as [n-n] Contract.

Inserting a Child Field in a Transaction Record

This operation inserts a child field in the selected Transaction Record.

To insert a child field in a transaction record

1. Navigate to the Imported Transaction Records screen.
2. Right-click on the Transaction Record. The operations menu then displays.
3. Choose Insert Child Field.

Inserting a Child Group in a Transaction Record

This operation inserts a child group in the selected Transaction Record.

To insert a child group in a transaction record

1. Navigate to the Imported Transaction Records screen.
2. Right-click on the Transaction Record. The operations menu then displays.
3. Choose Insert Child Group.

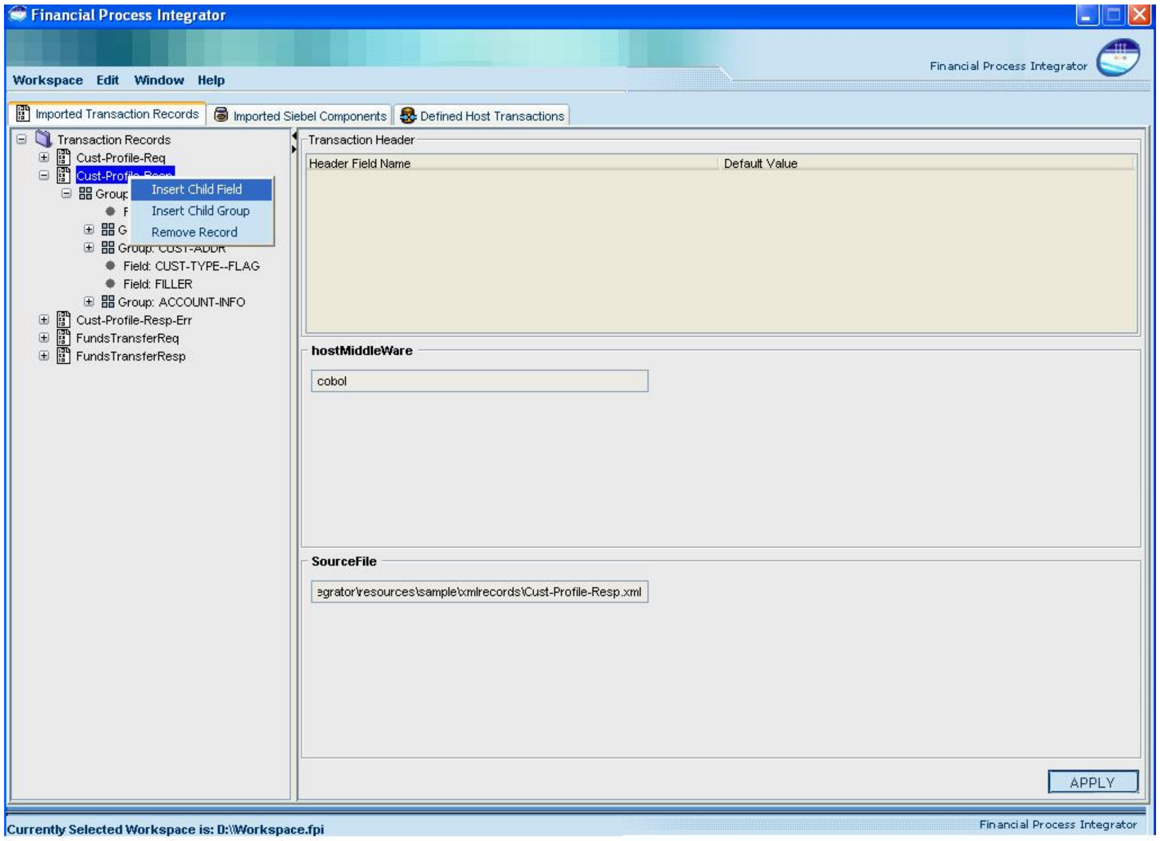
Removing a Child Group from a Transaction Record

This operation removes a child group from the selected Transaction Record.

To remove a child group from a transaction record

1. Navigate to the Imported Transaction Records screen.
2. Right-click on the Transaction Record. The operations menu then displays.
3. Choose Remove Record.

The following screen capture illustrates the structure of a sample Transaction Record and the options that are available when the user right-clicks on a Transaction Record.



hostMiddleWare: cobol

SourceFile: egrator\resources\sample\xmlrecords\Cust-Profile-Resp.xml

Currently Selected Workspace is: D:\\Workspace.fpi

Amending a Transaction Group

The following operations can be performed on a Transaction Group that has previously been imported:

- Insert a Field
- Insert a Group
- Insert a Child Field
- Insert a Child Group
- Remove a Group

Inserting a Field in a Transaction Group

This operation inserts a Field in the selected Transaction Group.

To insert a field in a transaction group

1. Navigate to the Imported Transaction Records screen.
2. Select the Transaction Group to which the Transaction Record belongs. The Transaction Groups then display.
3. Right-click on the Transaction Group. The operations menu then displays.
4. Choose Insert Field.

Inserting a Transaction Group

This operation inserts a Group after the selected Transaction Group to reflect the host transaction structure.

To insert a transaction group after an existing transaction group

1. Navigate to the Imported Transaction Records screen.
2. Select the Transaction Group which the new Transaction Group is to be inserted after.
3. Right-click on the Transaction Group. The operations menu then displays.
4. Choose Insert Group.

Inserting a Child Field within a Transaction Group

This operation inserts a Child Field within the selected Transaction Group.

To insert a child field within a transaction group

1. Navigate to the Imported Transaction Records screen.
2. Select the Transaction Group which the new Child Field is to be inserted within.
3. Right-click on the Transaction Group. The operations menu then displays.
4. Choose Insert Child Field.

Inserting a Child Group within a Transaction Group

This operation inserts a Child Group within the selected Transaction Group.

To insert a child group within a transaction group

1. Navigate to the Imported Transaction Records screen.
2. Select the Transaction Group which the new Child Group is to be inserted within.
3. Right-click on the Transaction Group. The operations menu then displays.

4 Choose Insert Child Group.

Removing a Transaction Group

This operation removes a Transaction Group

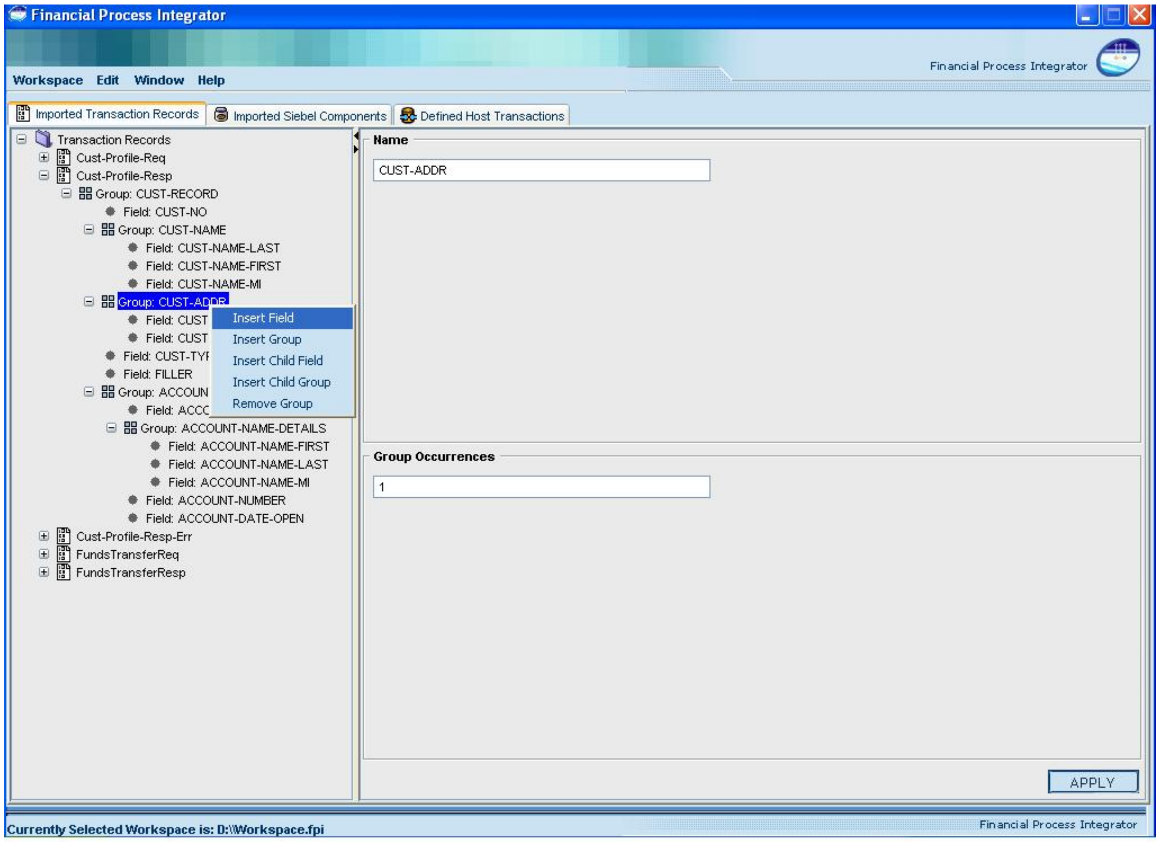
To remove a transaction group

1 Navigate to the Imported Transaction Records screen.

2 Right-click on the Transaction Group. The operations menu then displays.

3 Choose Remove Group.

The following screen capture illustrates the structure of a sample Transaction Record and the options that are available when the user right-clicks on a Transaction Group.


Financial Process Integrator
Workspace
Edit
Window
Help
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Insert Field
Insert Group
Insert Child Field
Insert Child Group
Remove Group
APPLY

```
Transaction Records
  Cust-Profile-Req
  Cust-Profile-Resp
    Group: CUST-RECORD
      Field: CUST-NO
      Group: CUST-NAME
        Field: CUST-NAME-LAST
        Field: CUST-NAME-FIRST
        Field: CUST-NAME-MI
      Group: CUST-ADDR
        Field: CUST
        Field: CUST
      Field: CUST-TYI
      Field: FILLER
      Group: ACCOUN
        Field: ACCC
        Group: ACCOUNT-NAME-DETAILS
          Field: ACCOUNT-NAME-FIRST
          Field: ACCOUNT-NAME-LAST
          Field: ACCOUNT-NAME-MI
        Field: ACCOUNT-NUMBER
        Field: ACCOUNT-DATE-OPEN
  Cust-Profile-Resp-Err
  FundsTransferReq
  FundsTransferResp
```

Name: CUST-ADDR

Group Occurrences: 1

Currently Selected Workspace is: D:\\Workspace.fpi

Amending a Transaction Field

The following operations can be performed on a Transaction Field that has previously been imported:

- Insert a Field
- Insert a Group
- Remove a Field

Inserting a Transaction Field

This operation inserts a Field after the selected Transaction Field.

To insert a field after an existing transaction field

1. Navigate to the Imported Transaction Records screen.
2. Select the Transaction Field which the new Field is to be inserted after.
3. Right-click on the Transaction Field. The operations menu then displays.
4. Choose Insert Field.

Inserting a Transaction Group

This operation inserts a Transaction Group after the selected Transaction Field.

To insert a transaction group after an existing transaction field

1. Navigate to the Imported Transaction Records screen.
2. Select the Transaction Field which the new Group is to be inserted after.
3. Right-click on the Transaction Field. The operations menu then displays.
4. Choose Insert Group.

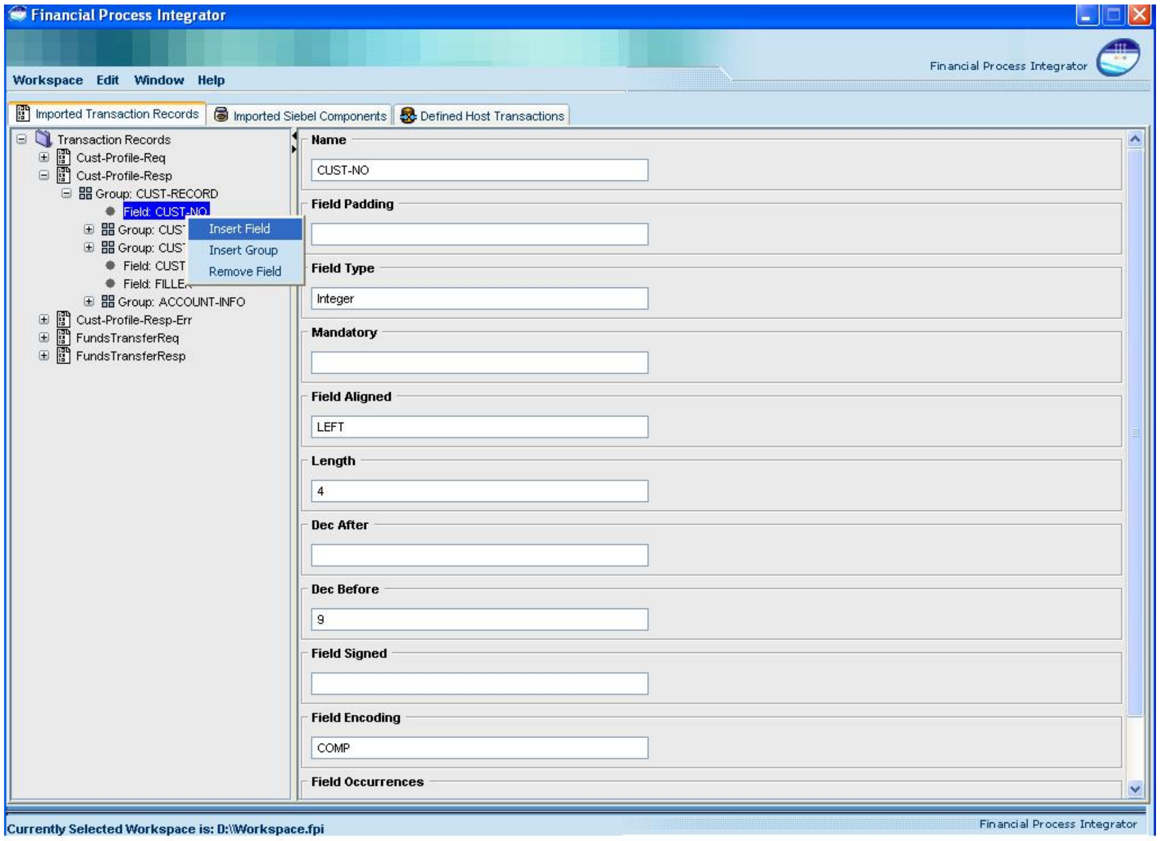
Removing a Transaction Field

This operation removes the selected Transaction Field.

To remove a transaction field

1. Navigate to the Imported Transaction Records screen.
2. Right-click on the Transaction Field which is to be removed. The operations menu then displays.
3. Choose Remove Field.

The following screen capture illustrates the structure of a sample Transaction Record and the options that are available when the user right-clicks on a Transaction Field.



Name: CUST-NO
Field Padding:
Field Type: Integer
Mandatory:
Field Aligned: LEFT
Length: 4
Dec After:
Dec Before: 9
Field Signed:
Field Encoding: COMP
Field Occurrences

Currently Selected Workspace is: D:\\Workspace.fpi

11 Financial Objects and Components

About the SRF Model

The Siebel Retail Finance Models are generated by the SRF Design Tools and comply with the SRF automated methodology. Financial Objects and Financial Components are imported via the SRF Model. Broker Objects are not imported in the model but are defined via the FPI GUI on the Imported Siebel Components tab.

When the SRF Model Financial Objects and Components are imported into the FPI they are displayed in the Siebel Components list on the Imported Siebel Components tab.

Financial Objects

A Financial Object represents an Entity EJB that is being used by the SRF application. Financial Objects must be imported in XML format. When Financial Objects are imported into the FPI they are listed under the Financial Objects list node on the Imported Siebel Components tab.

Financial Components

A Financial Component represents a Session EJB that is being used by the SRF application. Financial Objects must be imported in XML format. Currently, Financial Components are not used in the mapping of transaction metadata so this list may be empty. When Financial Components are imported into the FPI they are listed under the Financial Components list node on the Imported Siebel Components tab.

Importing an SRF Model

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

Re-Importing an SRF Model

To re-import an existing model into the workspace, to reflect changes that have been made to the model, import the model again as described above. The FPI tool allows a Model to change and be re-imported without having to re-do mapping information. If you have mapped an attribute belonging to a Financial Object to a host transaction field and this attribute has been renamed in the model you are importing, the mapping information is automatically updated to reflect this change.

For example, if a Financial Object Attribute is renamed in the model from CustomerId to CusId and you re-import the model the following will automatically happen:

- Any references in the mapping screens to CustomerId will automatically be updated to CusId.
- If you select the Customer Object from the FinancialObjectName combo box (in the mapping screen), the attributes corresponding to this Object will contain CusId and not CustomerId.
- If the CustomerId had been removed from the re-imported Model any references to the CustomerId attribute in the mapping screens would be changed to REMOVED.
- If the Customer Object was removed from the model any references to the object as well as the attribute would be changed to REMOVED.

NOTE: The FPI tool manages the updating of Financial Object names and their attribute names by utilizing the unique roseId that Rational Rose associates with each Object, Attribute and so on. Therefore, if you re-import a different model with the same Objects and Attributes this automatic synching with the Mapping screens will not work because Objects and Attributes in different Rose models will not have the same ID's.

The FPI tool also creates a report file when it is re-importing a model, this report file is named design-report.txt. This report file details any Financial objects or their attributes that have been renamed or deleted following an import.

Removing an SRF Model

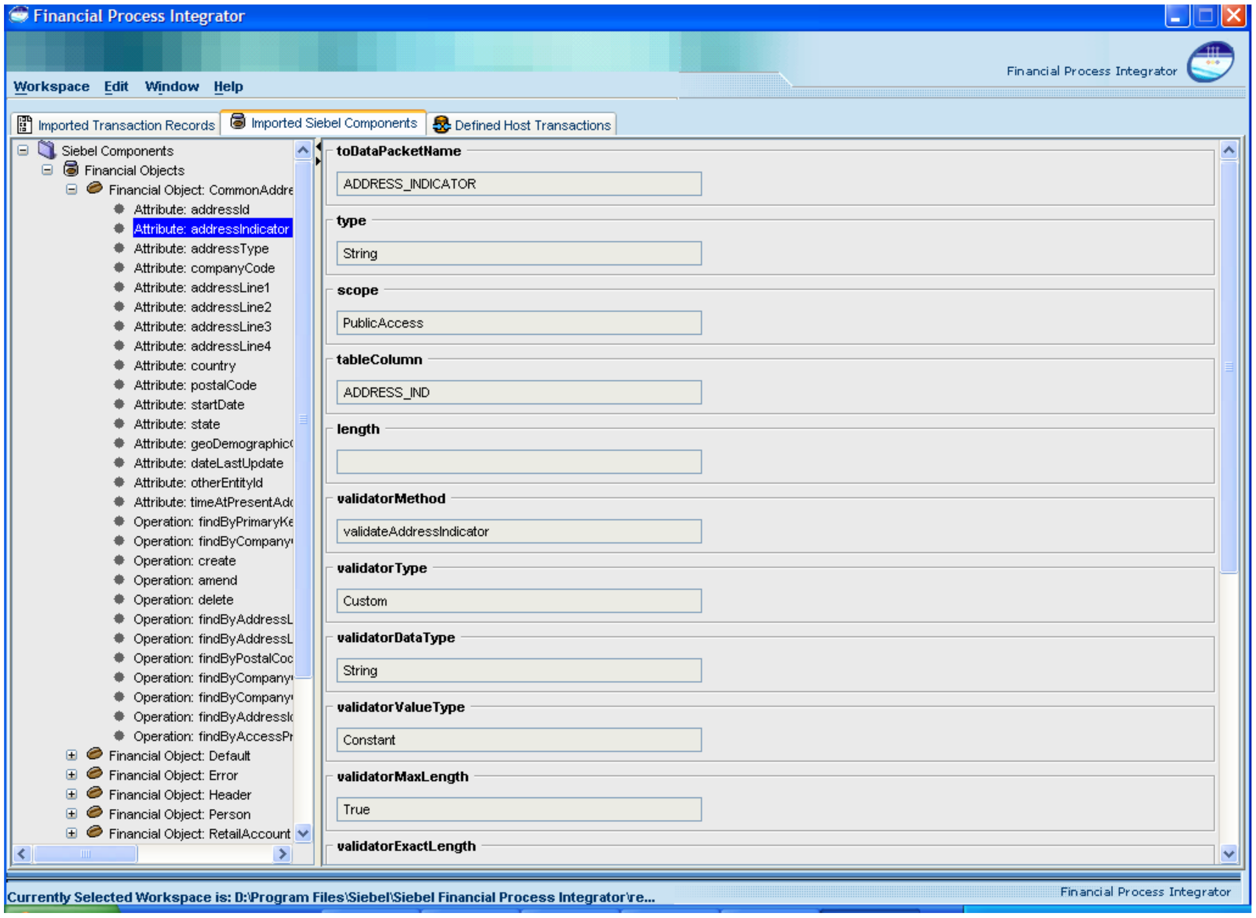
Selecting the Remove a Siebel Model menu item will remove all previously imported Financial Objects and Financial Components from your workspace. If a financial object that is part of a user-defined host transaction is removed, you will have to go back and re-map the fields in that host transaction to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

About Financial Object Properties

All Financial Objects, Attributes and Operations 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 Object, Attribute or Operation is selected. Financial Object properties are not editable in the FPI and can only be changed in the SRF Design Tools. Each Financial Object node has some of the following properties associated with it:

Property	Description
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. This property only applies to a CMP (Container Managed Persistence) scenario.
ToDataPacket Name	The DataPacket name that this Financial Object is associated with.

The following screen capture shows the structure of a sample Financial Object, its Attributes and Operations and sample Attribute properties:


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

- Siebel Components
  - Financial Objects
    - Financial Object: CommonAddre
      - Attribute: addressId
      - Attribute: addressIndicator
      - Attribute: addressType
      - Attribute: companyCode
      - Attribute: addressLine1
      - Attribute: addressLine2
      - Attribute: addressLine3
      - Attribute: addressLine4
      - Attribute: country
      - Attribute: postalCode
      - Attribute: startDate
      - Attribute: state
      - Attribute: geoDemographic
      - Attribute: dateLastUpdate
      - Attribute: otherEntityId
      - Attribute: timeAtPresentAd
      - Operation: findByPrimaryKe
      - Operation: findByCompany
      - Operation: create
      - Operation: amend
      - Operation: delete
      - Operation: findByAddressL
      - Operation: findByAddressL
      - Operation: findByPostalCo
      - Operation: findByCompany
      - Operation: findByCompany
      - Operation: findByAddressI
      - Operation: findByAccessPr
    - Financial Object: Default
    - Financial Object: Error
    - Financial Object: Header
    - Financial Object: Person
    - Financial Object: RetailAccount

toDataPacketName	ADDRESS_INDICATOR
type	String
scope	PublicAccess
tableColumn	ADDRESS_IND
length	
validatorMethod	validateAddressIndicator
validatorType	Custom
validatorDataType	String
validatorValueType	Constant
validatorMaxLength	True
validatorExactLength	

Currently Selected Workspace is: D:\Program Files\Siebel\Siebel Financial Process Integrator\re...

12 Broker Objects

About Broker Request and Response Objects

Broker Request and Response Objects are used to define parameters that do not map to the attributes in the Financial Objects. Broker Request and Response Objects are used in conjunction with the SessionAmendHelper and the TransactionHandlerBroker.

Creating Broker Request Objects

The main Broker Request Objects list node exists in the FPI. Broker Request Objects and their fields are not imported in the model and must be defined via the FPI GUI.

To create a Broker Request Object

1. Navigate to the Imported Siebel Components tab.
2. Right-click on the Broker Request Objects node. The operations menu is then launched.
3. Select Create Broker Request Object. The Add Broker Request Object dialog box displays.
4. Enter a name for the new Broker Request Object.
5. Click the Finish button.

The new Broker Request Object displays below the main Broker Request Objects node. Once the new Broker Request Object has been created the fields should be defined.

To create the Broker Request Object fields

1. Navigate to the Imported Siebel Components tab.
2. In the left pane Select the Broker Request Object that the fields are to be defined for.
3. Right-click on the workspace on the right and select Add. A row is crested for the new field.
4. Enter a name for the new field.
5. Repeat steps 3 and 4 for each field to be created.

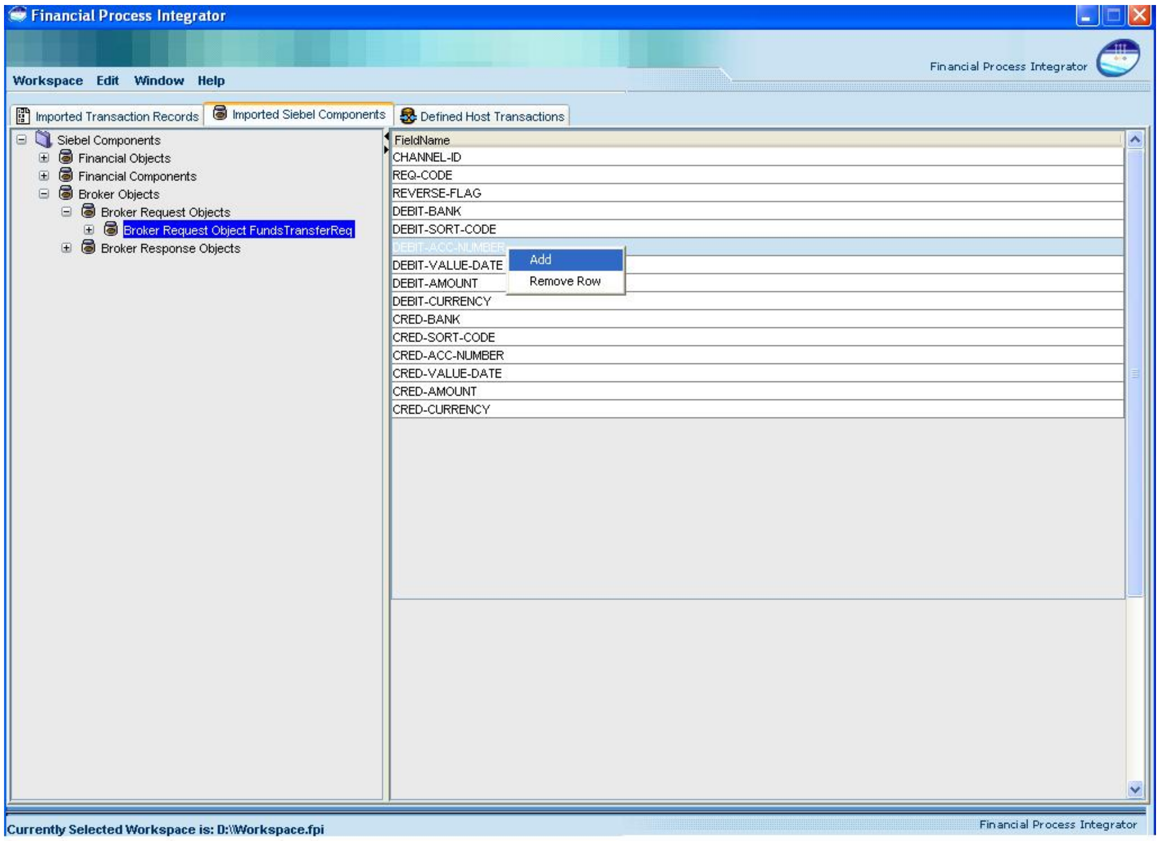
Removing Broker Request Objects

This functionality enables the removal of a Request Broker Object and its fields.

To remove a Broker Request Object

1. Navigate to the Imported Siebel Components tab.
2. Right-click on the Broker Request Object that is to be removed. The operations menu then displays.
3. Select Delete Broker Request Object.

The following screen capture illustrates the structure of a sample Broker Request Object:


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

Siebel Components
- Financial Objects
- Financial Components
- Broker Objects
  - Broker Request Objects
    - Broker Request Object FundsTransferReq
  - Broker Response Objects

FieldName
CHANNEL-ID
REQ-CODE
REVERSE-FLAG
DEBIT-BANK
DEBIT-SORT-CODE
DEBIT-ACC-NUMBER
DEBIT-VALUE-DATE
DEBIT-AMOUNT
DEBIT-CURRENCY
CRED-BANK
CRED-SORT-CODE
CRED-ACC-NUMBER
CRED-VALUE-DATE
CRED-AMOUNT
CRED-CURRENCY

Currently Selected Workspace is: D:\\Workspace.fpi

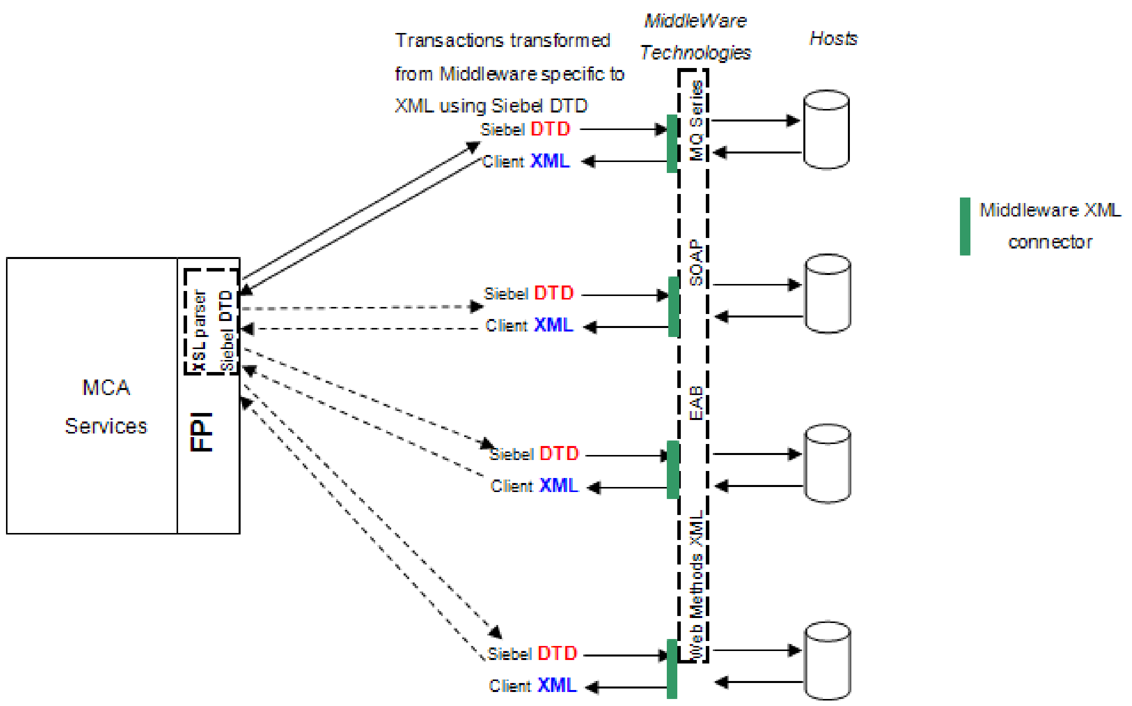
Creating and Removing Broker Response Objects

The procedures for creating a Broker Response Object, adding Broker Response Object fields and removing a Broker Response Object are the same as those detailed above for Broker Request Object except that you select the Broker Response Objects node rather than the Broker Request Objects node.

13 XML Transaction Record Example

In this example we import an XML Transaction Record. The underlying middleware type of this host transacton is Cobol.

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" name="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 name="Account_Info" groupOccurrences="2" redefines="">

                <GroupOverview>This returns the Account information</GroupOverview>

<Field fieldOccurrences="1" name="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" name="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 name="Address_Details" groupOccurrences="2" redefines="">

                <GroupOverview>This returns the Account information</GroupOverview>

<Field fieldOccurrences="3" name="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" name="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" name="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 name="Account_Info_All" groupOccurrences="1" redefines="">

                <Group name="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" name="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 name="Account_Info_All_2" groupOccurrences="1"     redefines="Account_Info">

<GroupOverview>This returns the Account information in a different format</GroupOverview>

<Field fieldOccurrences="1" name="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" name="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>
```

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/Model` directory.

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

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'.

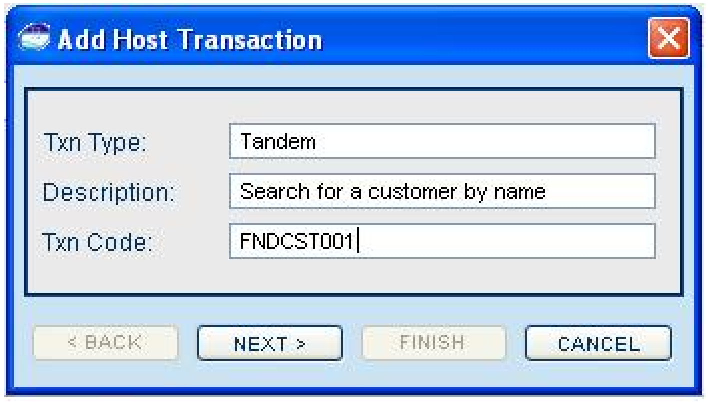
We're almost ready to add a host transaction but first we have to import a host connector.

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.

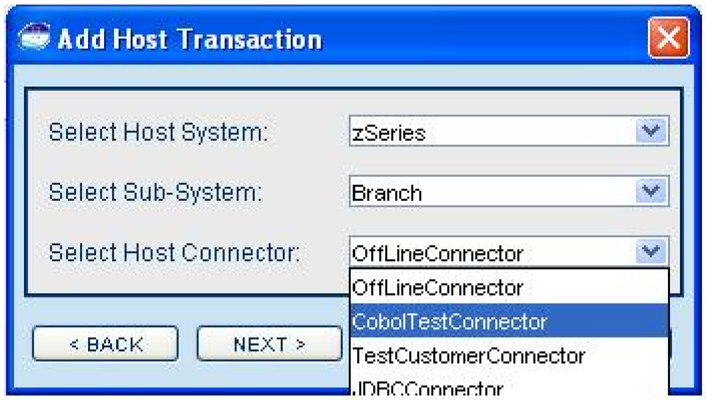
Creating a Host Transaction

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:



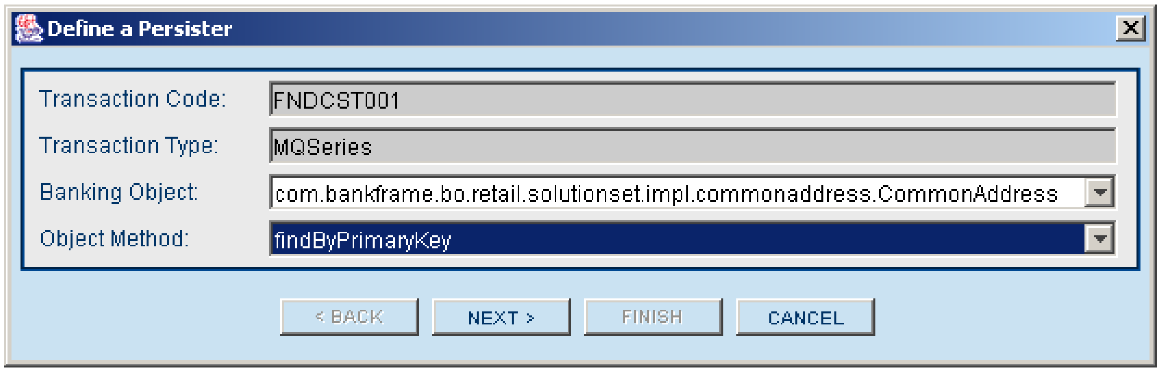
Add Host Transaction

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:



Define a Persister

Transaction Code: FNDCST001

Transaction Type: MQSeries

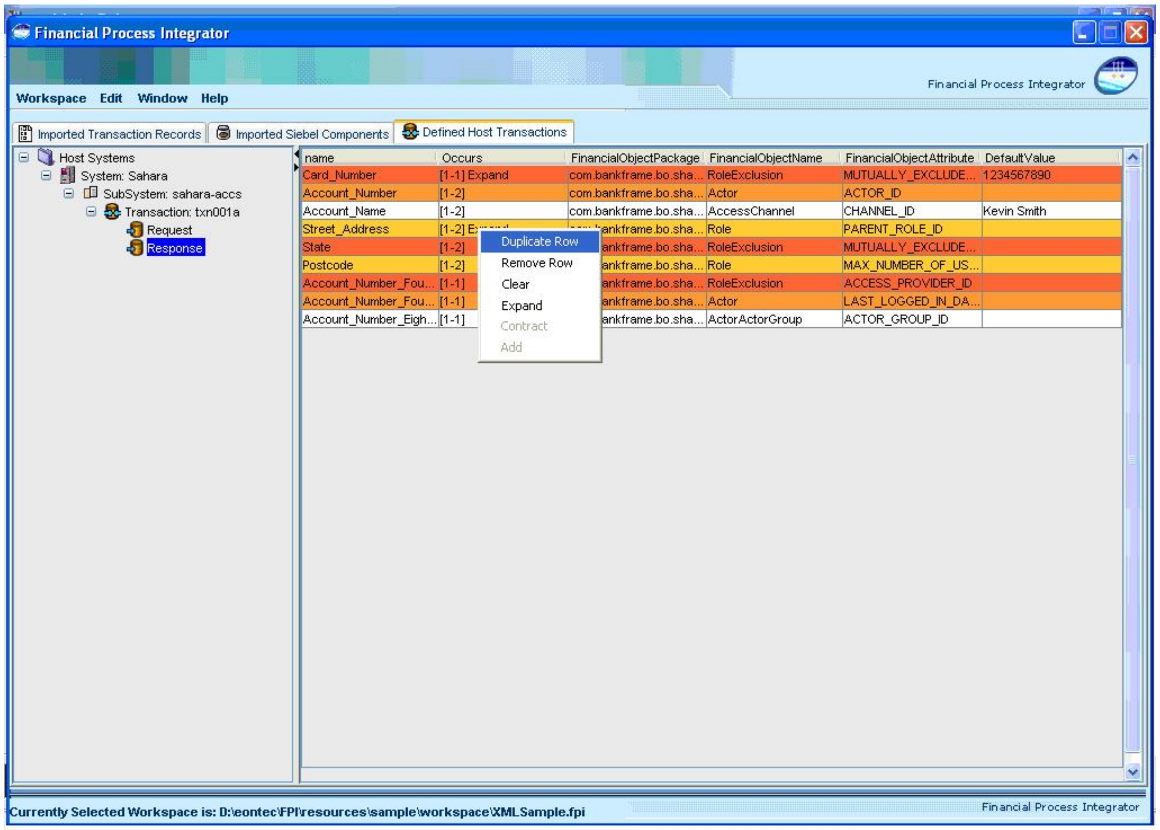
Banking Object: com.bankframe.bo.retail.solutionset.impl.commonaddress.CommonAddress

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.

Mapping Fields to Attributes

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

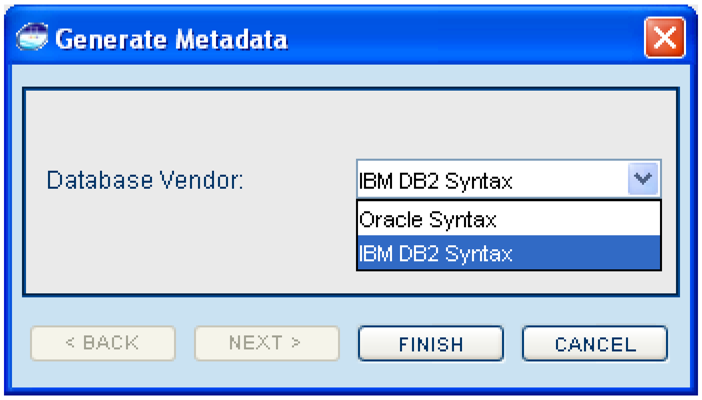

Financial Process Integrator
Workspace
Edit
Window
Help
Financial Process Integrator
Imported Transaction Records
Imported Siebel Components
Defined Host Transactions
Host Systems
System: Sahara
SubSystem: sahara-accs
Transaction: txn001a
Request
Response
Duplicate Row
Remove Row
Clear
Expand
Contract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

Generating Metadata

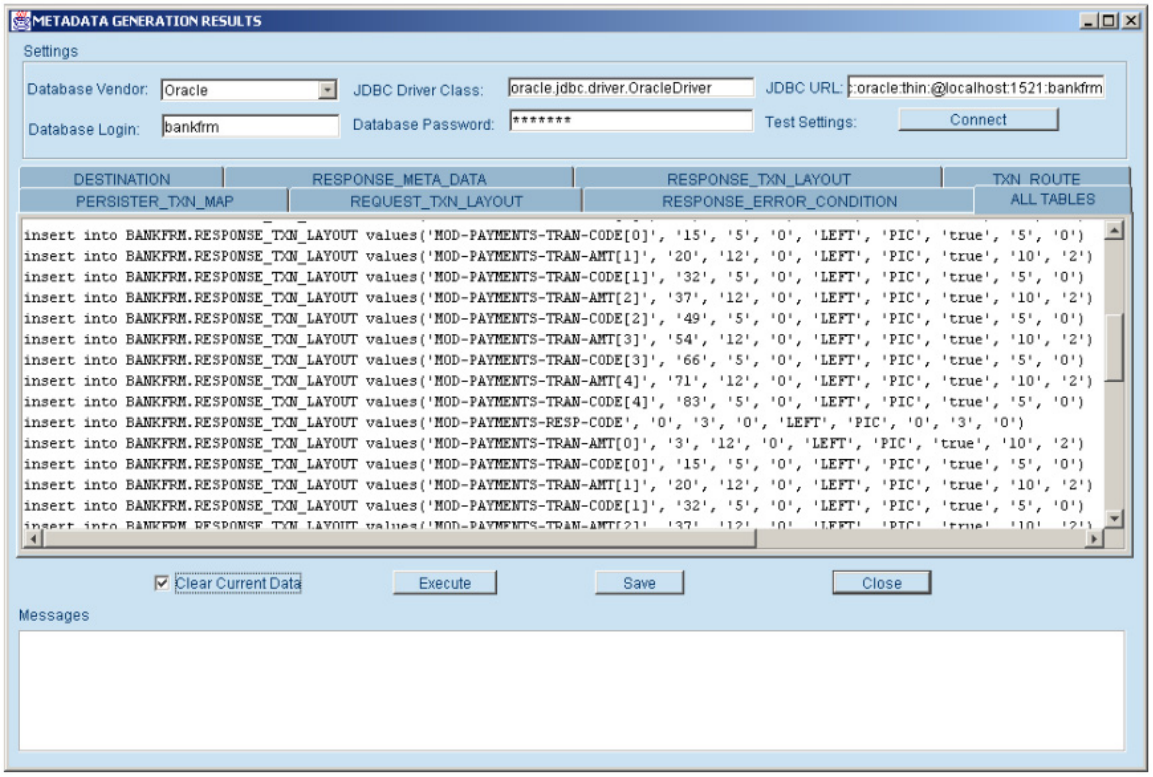
Click on the Workspace menu item in the application window, and select the 'Generate SQL' menu item. This will bring up a wizard that prompts you to choose a metadata format, and a database vendor to generate the metadata.



Database Vendor: IBM DB2 Syntax

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:



```
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')
```

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.

14 Refreshing MetaData

About Refreshing MetaData

If new user-defined columns are added to the RESPONSE_META_DATA table the Refresh Meta Data utility identifies these columns, updates the BankFrameConstants.properties file and the MetaData mappings in the applicable Response nodes of the corresponding Host Transaction on the Defined Host Transactions tab. When the MetaData SQL is then regenerated (by selecting Workspace > Generate SQL on the main FPI menu) the new user-defined columns are included in the SQL.

Refreshing MetaData

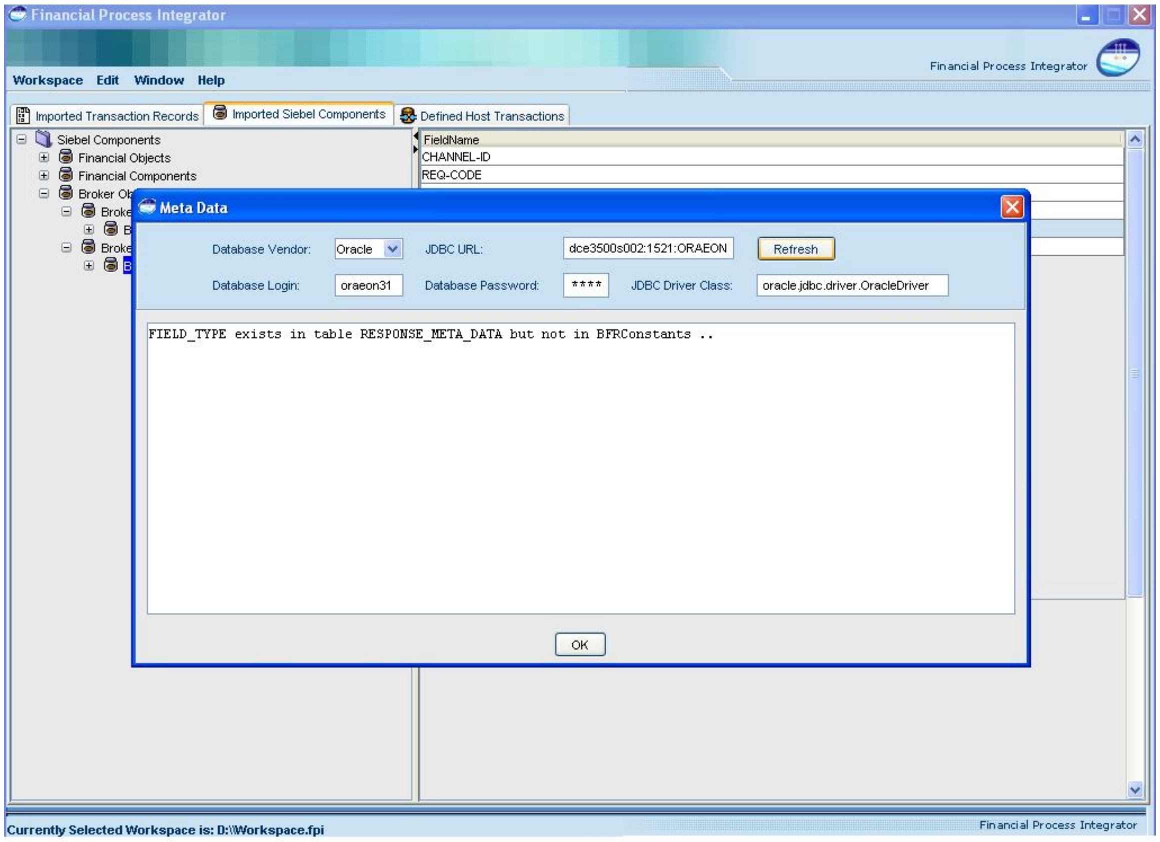
The Refresh Meta Data utility looks for any new user-defined fields that are in the RESPONSE_META_DATA table but are not in the BankFrameConstants.properties file.

To refresh the MetaData with user-defined changes:

1. Navigate to the Window > Refresh Meta Data > Meta Data screen.
2. Select the Database Vendor from the dropdown menu.
3. Enter the JDBC URL, Database Login, Database Password and JDBC Driver Class.
4. Click the Refresh button.

The tool then displays a message listing any field that exists in the RESPONSE_META_DATA table but does not exist in the BankFrameConstants.properties file, updates BankFrameConstants.properties and updates the mappings in the Defined Host Transactions tab.

The screen capture below illustrates the Refresh MetaData utility informing the user that an extra column called FIELD_TYPE exists in the RESPONSE_META_DATA table but not in BankFrameConstants.properties.



Database Vendor: Oracle
JDBC URL: dce3500s002:1521:ORAEON
Database Login: oraeon31
Database Password: ****
JDBC Driver Class: oracle.jdbc.driver.OracleDriver

```
FIELD_TYPE exists in table RESPONSE_META_DATA but not in BFRConstants ..
```

Currently Selected Workspace is: D:\\Workspace.fpi

15 Cobol Copybook Conversion

About the Cobol Copybook Converter

Transaction Records must be in XML format, compliant with the FPITransaction DTD, before being imported into the Financial Process Integrator tool. If the existing Transaction Records are in Cobol Copybook format they need to be converted to FPITransaction.dtd format before being imported.

The command line utility run.bat is provided to help convert a Cobol Copybook to FPITransaction.dtd format. This file is located in the `<%FPI Installation%>\resources\copybookconverter` folder. Sample Cobol Copybooks are also provided in the `<%FPI Installation%>\resources\sample\` copybookrecords folder.

The copybook converter run.bat file takes two arguments:

- The name of and path to the input file (the cobol copybook).
- The name of and path to the output XML file.

A period must exist at the end of each line in the Cobol Copybook for correct conversion.

Converting Cobol Copybooks to FPITransaction DTD Format

To convert a Cobol Copybook to FPITransaction DTD format, with the above limitations, complete the steps below.

To convert a convert a Cobol Copybook to FPITransaction DTD format

1. Open a DOS window (Select Start > Run and type cmd in the Open field, click OK).
2. Type cd `<FPI-Install-Directory>\resources\copybookconverter\` at the DOS prompt, where `<FPI-Install-Directory>` is the path to the FPI install directory.
3. Type `Run <CopyBookName> <OutputFile>`, where `<CopyBookName>` is the name of and path (relative or absolute) to the Cobol Copybook file and `<CopyBookName>` is the name of and path to the output XML file that the utility will create.

16 Configuring FPI Settings

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

Key	Sample Value	Description
		statements. e.g. insert into BANKFRM.REQUEST_TXN_LAYOUT 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 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.

18 FPI Meta Data Rules

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

property	Description	Populated	Editable	Valid Input	Used In
*IsSigned	Is Field Signed?	On import of cobol	No	Boolean as String (true, false)	request_txn_layout, response_txn_layout
NumBefore	Number of spaces before decimal (Numeric data only)	On import of cobol	No	Integer as String	request_txn_layout, 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

property	Description	Populated	Editable	Valid Input	Used In
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
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

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

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

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 properties for this connector	By user after import of Connectors	Yes	Alphanumeric as String (key-value property pairs are semi-colon delimited)	destination

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

Host Transaction Fields

Property	Description	Populated	Editable	Valid Input	Used In
TransactionCode	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
TransactionType	Type of this transaction	By user on creation of transaction	Yes	Alphanumeric as String	request_txn_layout, response_meta

Property	Description	Populated	Editable	Valid Input	Used In
		transaction			_data, response_error_condition, persister_txn_map, txn_route
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
SubSystemName	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

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
HostConnectorNam	Cache Policy	Drop Down from list of	Yes	String chosen from set list	persister_txn_m ap

Property	Description	Populated	Editable	Valid Input	Used In
ectorNam e		from list of imported connectors		from set list	ap (Cache_Policy)

Request Mapping Fields

Property	Description	Populated	Editable	Valid Input	Used In
Txn Field	Name of the host record field for this transaction	From selected Transactio n Record	No	Alphanumeric as String	request_txn_lay out
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_lay out 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_lay out

Response Mapping Fields

Property	Description	Populated	Editable	Valid Input	Used In
Txn Field	Name of the host record field for this transaction	From selected Transactio n Record	No	Alphanumeric as String	request_txn_lay out

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 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