TIBCO ActiveMatrix BusinessWorks™ Plug-in for SWIFT User's Guide

Software Release 6.1
April 2016

Two-Second Advantage®

Important Information

Contents

TIBCO Documentation and Support Services

Documentation for this and other TIBCO products is available on the TIBCO Documentation site. This site is updated more frequently than any documentation that might be included with the product. To ensure that you are accessing the latest available help topics, please visit:

https://docs.tibco.com

Product-Specific Documentation

Documentation for TIBCO products is not bundled with the software. Instead, it is available on the TIBCO Documentation site at https://docs.tibco.com/products/tibco-activematrix-businessworks-plug-in-for-swift. To directly access documentation for this product, double-click the following file:

`TIBCO_HOME/release_notes/TIB_bwpluginswift_version_docinfo.html`

where *TIBCO_HOME* is the top-level directory in which TIBCO products are installed. On Windows, the default *TIBCO_HOME* is `C:\Program Files\tibco`. On UNIX systems, the default *TIBCO_HOME* is `/opt/tibco`.

The following documents for this product can be found on the TIBCO Documentation site:

- *TIBCO ActiveMatrix BusinessWorks Plug-in for SWIFT Installation*
- *TIBCO ActiveMatrix BusinessWorks Plug-in for SWIFT User's Guide*
- *TIBCO ActiveMatrix BusinessWorks Plug-in for SWIFT Examples*
- *TIBCO ActiveMatrix BusinessWorks Plug-in for SWIFT Release Notes*

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support:

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

How to Join TIBCO Community

TIBCO Community is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCO Community offers forums, blogs, and access to a variety of resources. To register, go to the following web address:

https://community.tibco.com

Product Overview

TIBCO ActiveMatrix BusinessWorks™ Plug-in for SWIFT provides the functionalities to render, parse, and validate MT and MX messages, as well as route MT messages to different activities based on different message type.

TIBCO ActiveMatrix BusinessWorks Plug-in for SWIFT contains a SWIFT MT palette and a SWIFT MX palette. The activities in the SWIFT MT palette are used to handle the MT messages, and the activities in the SWIFT MX palette are used to handle the MX messages.

The following activities are available in the SWIFT MT palette:

- Parse SWIFT MT Activity

  You can use this activity to validate an MT message and parse it to XML format.
- Render SWIFT MT Activity

  You can use this activity to render an MT message from XML format to the MT message format.
- Route SWIFT MT Activity

  You can use this activity to route a message to different activities based on the message types.
- Generate SWIFT BICPlusIBAN Activity

  You can use this activity to generate an IBAN.
- Validate SWIFT BICPlusIBAN Activity

  You can use this activity to validate an IBAN.

Before parsing or rendering an MT message, you have to load the corresponding message type schema. See Load SWIFT MT Schema Shared Resource for more information.

The following activities are available in the SWIFT MX palette:

- Parse SWIFT MX Activity

  You can use this activity to validate an MX message and parse it to XML format.
- Render SWIFT MX Activity

  You can use this activity to generate specific MX message.

Before parsing or rendering an MX message, you have to load the corresponding message type schema. See Load SWIFT MX Schema Shared Resource for more information.

SWIFT Overview

Society for Worldwide Interbank Financial Telecommunication (SWIFT) is an organization that provides an automated fast and safe means of sending and receiving financial messages between financial institutions worldwide.

SWIFT users include financial institutions such as banks, brokers, dealers, or investment managers. Typical users of SWIFT deal with payments, securities, foreign exchange, money markets, treasury, and trade. All SWIFT users are identified in the SWIFT network by a unique address called a Bank Identifier Code (BIC). To exchange messages over the SWIFT network, every user must have at least one unique BIC.

SWIFT processes information such as data, texts, or commands in the form of messages. SWIFT provides the following two applications which make all messaging functions and facilities available to users:

- General Purpose Application (GPA)

  Controls how users communicate within SWIFT.

- Financial Application (MT)

  Controls the user-to-user messaging facilities within SWIFT.

SWIFT Messages

SWIFT messages mainly include MT messages and MX messages. A SWIFT message must conform to SWIFT standards and structure requirements.

MT Message

The SWIFT MT messages start with the letters MT followed by a 3-digit number. The first digit represents the message category, the second digit represents a group of related parts in a transaction life cycle, and the third digit represents the message type.

MT Message Types

SWIFT MT messages mainly include the following types:

System Messages

Sent by a user to the SWIFT system (delivery notifications, retrievals) or the other way around (retrieved messages, non-delivery warnings). System messages are represented by `MT0`*nn*, and used for system-level information, such as delivery notifications and non-delivery warnings.

User to User Messages

Enable financial transactions and are sent from one user to another. These messages are represented by `MT1`*nn* through `MT9`*nn*.

Service Messages

Also known as control messages and related either to system commands `LOGIN`, `SELECT`, or `QUIT` or to message acknowledgments. For example, when you send a message to the SWIFT network, SWIFT accepts the message and sends you an ACK if the syntax of the message is correct. If not, it returns a NAK.

MT Message Structure

All SWIFT MT messages are ASCII text messages that have the following structure:

- Basic header block

  This block is represented by `{1:}` and includes details, such as the application ID, service ID, address of the logical terminal, session number, sequence number, and so on. The application ID in this block helps you identify whether a message is a GPA message (system message) or an MT message (user to user message). For example, `'M'` indicates that the message is an MT message and `'A'` indicates that the message is a GPA message.

- Application header block

  This block is represented by `{2:}`. Application headers contain two types: input and output. The structure of the block varies depending on the type of the application header. This header block typically includes details, such as the message type, message priority, delivery monitoring, and so on.

- User header block

  This block is represented by `{3:}` and includes details, such as the banking priority code and so on. This is an optional block.

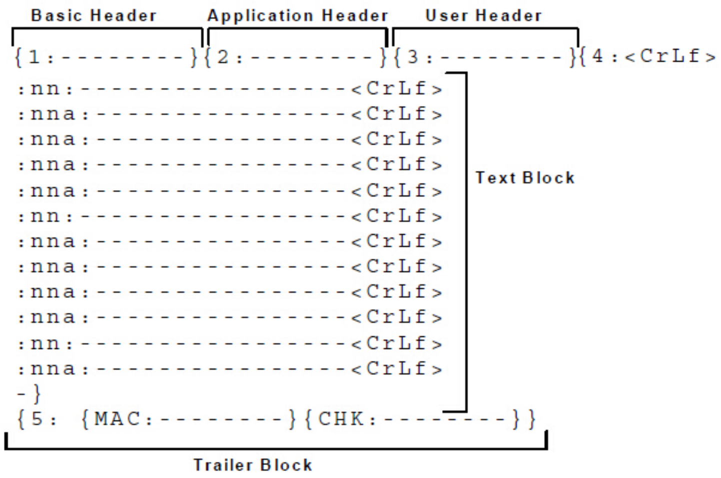
- Text block

  This block is represented by `{4:}` and contains the actual MT*nnn* message. This block includes details, such as the ordering customer, beneficiary customer, amount, currency code, date, and so on. This block consists of field tags of the format :nna: where nn is a number and a is an optional letter, which might be present on selected tags. The symbol CrLf is a control character and it represents Carriage Return/Line Feed. The symbol CrLf is a mandatory delimiter in this block.

- Trailer block

  This block is represented by `{5:}` and a message always ends in a trailer block. It is used for control purposes and includes details, such as message authentication code and checksum calculated for all the message types.

The following figure is a graphical representation of a typical SWIFT MT message:



```
{1:--------}{2:--------}{3:--------}{4:<CrLf>
:nn:----------------<CrLf>
:nna:----------------<CrLf>
:nna:----------------<CrLf>
:nna:----------------<CrLf>
:nna:----------------<CrLf>
:nn:----------------<CrLf>
:nna:----------------<CrLf>
:nna:----------------<CrLf>
:nna:----------------<CrLf>
:nna:----------------<CrLf>
:nn:----------------<CrLf>
:nna:----------------<CrLf>
-}
{5: {MAC:--------}{CHK:--------}}
```

MX Message

SWIFT MX message includes the following types:

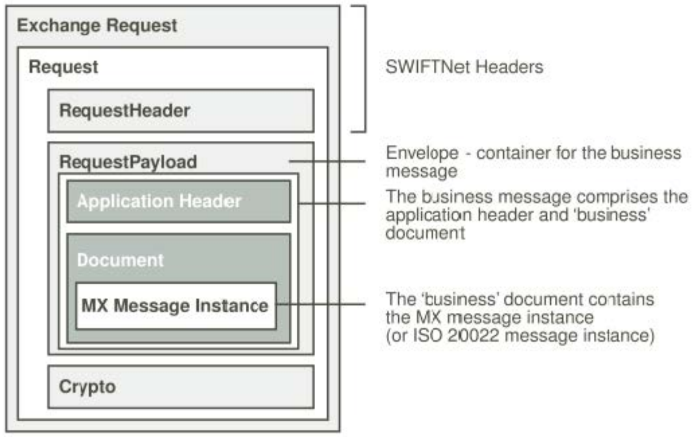
InterAct messages

These MX messages are used by the InterAct messaging service, and also by the key market infrastructures around the world. The InterAct messaging service is used for exchanging XML based financial messages and data between users. The structure of the InterAct messages mainly includes the transport, header, and document.

SAA messages

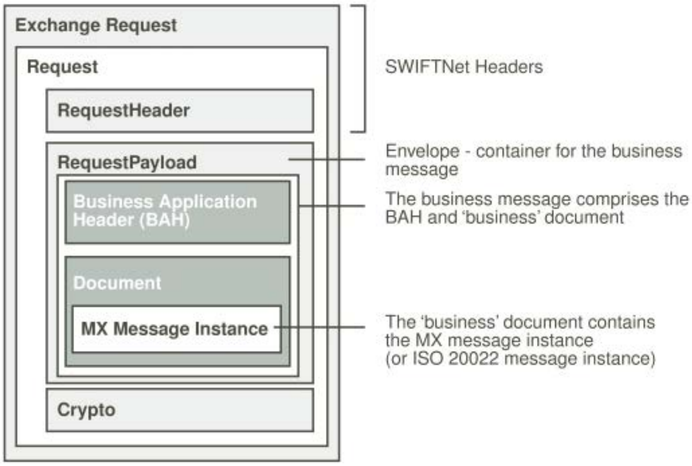
These messages are MX messages delivered to SWIFT Alliance Access (SAA). The structure of the SAA messages mainly includes the transport, header, and document.

The following figure shows the structure of the InterAct MX message blocks with Application Header:



Exchange Request
Request
RequestHeader
RequestPayload
Application Header
Document
MX Message Instance
Crypto
SWIFTNet Headers
Envelope - container for the business message
The business message comprises the application header and 'business' document
The 'business' document contains the MX message instance (or ISO 20022 message instance)

The following figure shows the structure of the InterAct MX message blocks with Business Application Header:


Exchange Request
Request
RequestHeader
RequestPayload
Business Application Header (BAH)
Document
MX Message Instance
Crypto
SWIFTNet Headers
Envelope - container for the business message
The business message comprises the BAH and 'business' document
The 'business' document contains the MX message instance (or ISO 20022 message instance)

Basic Message Flow

The basic SWIFT message flow involves syntax validation and confirmation of receipt and acceptance.

When you send a message to the SWIFT network, SWIFT validates the syntax of the message. If the message is correct, SWIFT accepts the message, sends you an ACK, and attempts to deliver the message to the receiver. If the message does not comply with the standards, SWIFT rejects it and returns a NAK. The NAK contains an error code, which helps the sender identify the type of error and its location. The sender can then correct the message before resending it to SWIFT.

SWIFT delivers the message as soon as the receiver is logging in to the SWIFT network. The interface at the receiver's end automatically confirms the receipt and acceptance of the message by sending a UAK. When a UAK is received by SWIFT, the message is considered to be delivered. If the message received is corrupted, the interface of the receiver sends a UNK to SWIFT, and SWIFT attempts to redeliver the message.

Communicating with the SWIFT Network

TIBCO ActiveMatrix BusinessWorks Plug-in for SWIFT does not communicate directly with the SWIFT network. To communicate with the SWIFT network, you must have SWIFT Alliance Access (SAA), SWIFT Alliance Entry (SAE), or any other interface approved by SWIFT.

Messages can be produced on a variety of user applications, but they can only be sent to SWIFT through SAA or SAE. Similarly, messages can be processed on a variety of user applications, but can only be received through SAA or SAE.

You can communicate with SAA using one of the following interfaces:

- CASmf

  CASmf is a software used for communication between SAA and other user applications. CASmf uses a MAPID, a name given to each instance of the messaging application, to establish the communication between the user application and SAA. The MAPID is defined in the user's host and in the SWIFT interface.

  CASmf also provides APIs to developers of the user application. With APIs, the user host environment can communicate with the SWIFT interface, establishing a real-time session. After a real-time session is established, financial messages can be exchanged. APIs can open, close, or abort a session, and send or receive data. The CASmf software uses TCP/IP as the communication protocol.

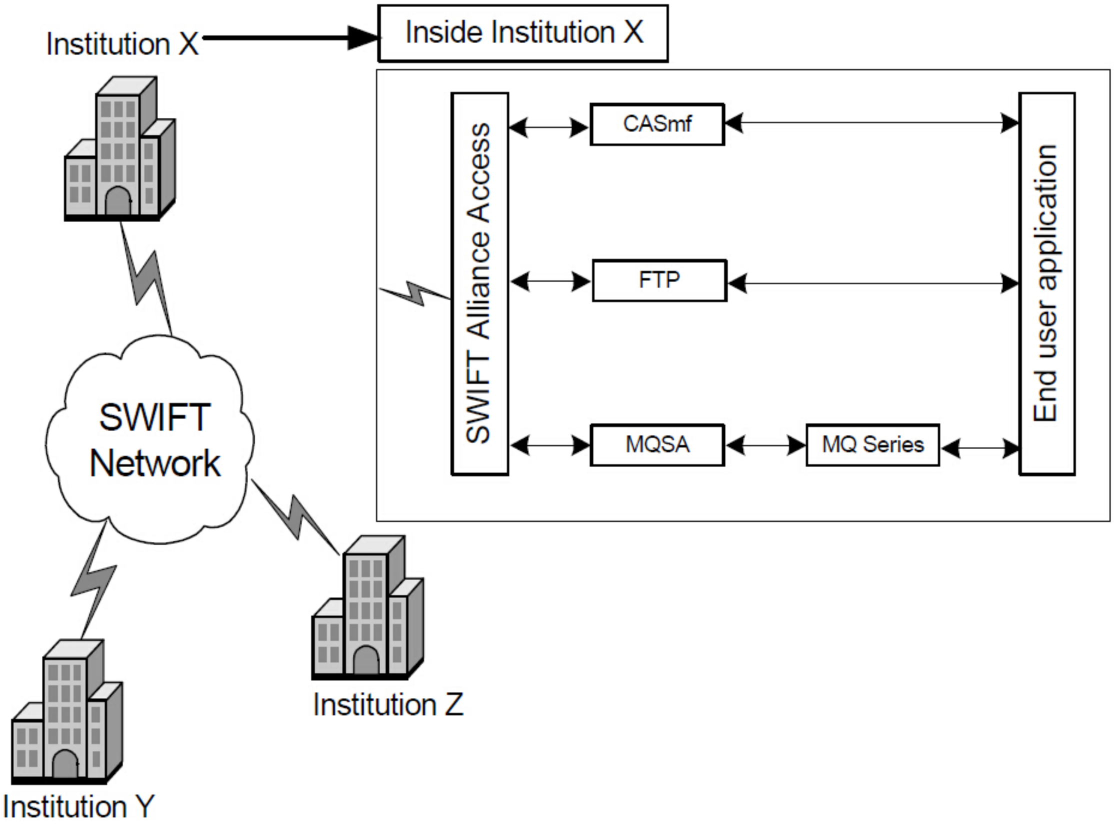
- FTP

An FTP server can be used for communication between SWIFT interfaces and user applications. The user application transfers SWIFT messages to a specified directory on the server. SAA picks up the SWIFT messages from this directory, and sends them to the SWIFT network.

- MQSA

  MQSA is a software used for communication between SAA and MQSeries. The MQSA interface is based on the SWIFT Alliance Development Kit (ADK). It uses ADK functions to communicate with SWIFT Alliance, and MQSeries functions to access message queuing services.

The following figure shows how the preceding interfaces are used to integrate SAA with the user application:



Terminology and Acronyms

The following terminology and acronyms are used in this manual:

Terminologies

Terminology	Meaning
ADK	Alliance Development Kit. ADK is a SWIFT product that developers of third-parties and financial institutions use to build their own applications for Alliance Access.

Terminology	Meaning
BIC	Business Identifier Code. BIC is an international standard for identification of institutions within the financial services industry. BICs are used in automated processing. They unambiguously identify a financial institution or a non-financial institution. The ISO 9362 standard specifies the elements and the structure of a BIC. A BIC consists of either eight or eleven contiguous characters. These characters comprise either the first three, or all four, of the following components: party prefix, country code, party suffix, and branch identifier. The International Organization for Standardization has designated SWIFT as the BIC registration authority.
Delivery Report	The SWIFT network can, either on its own or in response to a request from the user, report the delivery status of a user message. Typically, this message specifies whether the message was successfully delivered, could not be delivered, was acknowledged or not acknowledged (ACK or NACK) by the receiver or sender. Such a message is called a Delivery Report.
MT	A traditional message type for use on the SWIFT network.
MX	An XML message definition for use on the SWIFT network.
SAA	SWIFT Alliance Access. SAA is a prime multi-platform messaging interface designed to connect business applications to SWIFT messaging services. Customers can use SAA to connect single or multiple destinations to SWIFT with the maximum automation of system management tasks.
SAE	SWIFT Alliance Entry. SAE is a messaging interface that SWIFT develops for low-volume customers that use a single destination. SAE offers the flexibility and control of a private infrastructure.
SWIFT	Society for Worldwide Interbank Financial Telecommunication. This organization maintains a global store and forward network for secure financial messaging. In addition, it defines standard formats for interbank messages.
Transmission Report	The transmission report is a message that states whether the user message was accepted or not (ACK or NACK) by the network. Since the SWIFT network is a store and forward network, every user message is stored by the network before being forwarded to the receiver. At the time of storing, the network also validates the message.

Acronyms

Acronyms	Spelled-out Form
ACK	Positive Acknowledgment
API	Application Programming Interface
BEI	Business Entity Identifier

Acronyms	Spelled-out Form
IBAN	International Bank Account Number
JMS	Java Messaging Service
JNI	Java Native Interface
LTA	Logical Terminal Address
NAK	Negative Acknowledgment
UAK	User Positive Acknowledgment
UNK	User Negative Acknowledgment

Getting Started

This tutorial is designed for the beginners who want to use TIBCO ActiveMatrix BusinessWorks Plug-in for SWIFT in TIBCO Business Studio.

All the operations are performed in TIBCO Business Studio. See TIBCO Business Studio Overview to get familiar with TIBCO Business Studio.

The following procedure takes the Parse SWIFT MT activity as an example to demonstrate how to parse an MT message.

A basic procedure of using TIBCO ActiveMatrix BusinessWorks Plug-in for SWIFT includes:

1. Creating a Project
2. Creating a Load SWIFT MT Schema Shared Resource
3. Configuring a Process
4. Testing a Process
5. Deploying a Process

Creating a Project

The first task using the plug-in is creating a project. After creating a project, you can add resources and processes.

An Eclipse project is an application module configured for TIBCO ActiveMatrix BusinessWorks. An application module is the smallest unit of resources that is named, versioned, and packaged as part of an application.

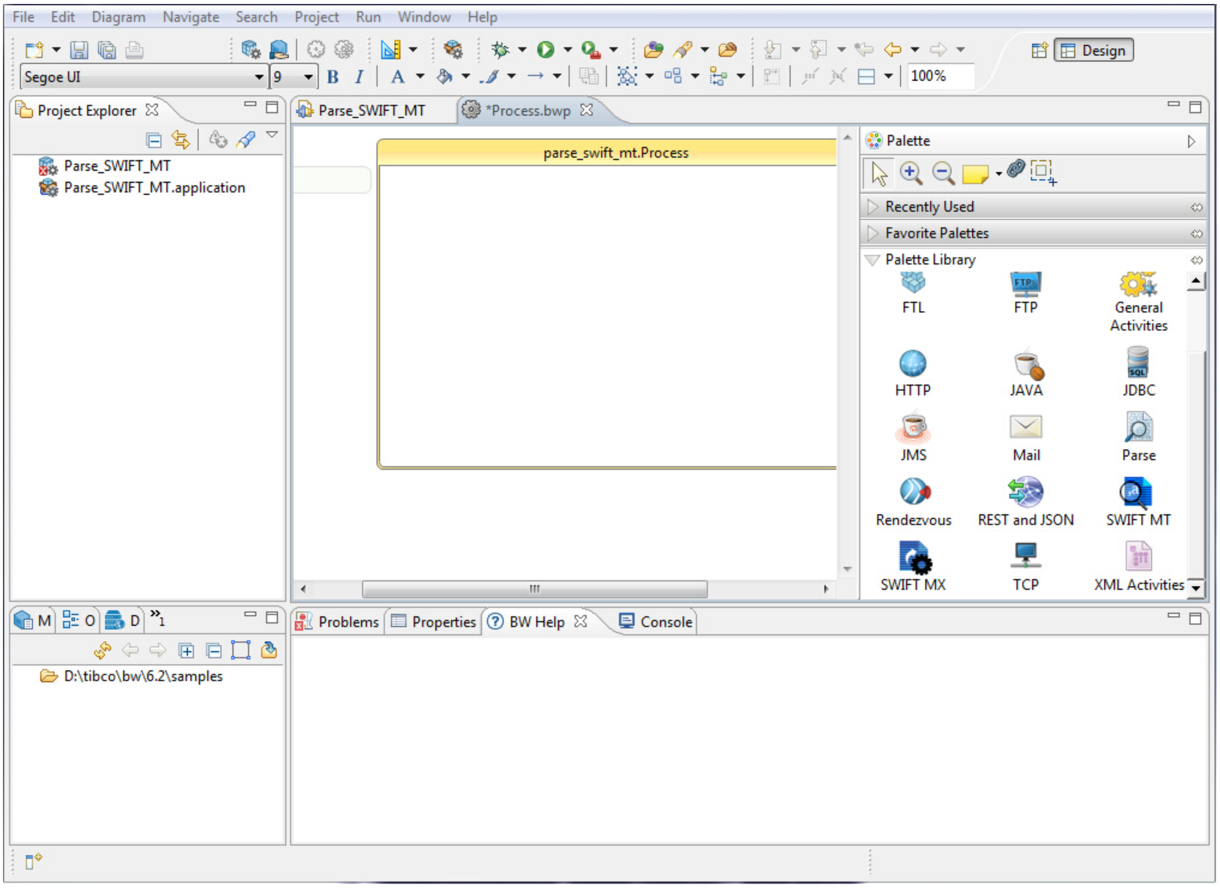
Procedure

1. Start TIBCO Business Studio using one of the following ways:
   - Microsoft Windows: click **Start** > **All Programs** > **TIBCO** > *TIBCO_HOME* > **TIBCO Business Studio** ***version_number*** > **Studio for Designers**.
   - Mac OS and Linux: run the TIBCO Business Studio executable file located in the `TIBCO_HOME/studio/version_number/eclipse` directory.
2. From the menu, click **File** > **New** > **BusinessWorks Resources** to open the BusinessWorks Resource wizard.
3. In the Select a wizard dialog, click **BusinessWorks Application Module** and click **Next** to open the New BusinessWorks Application Module wizard.
4. In the Project dialog, configure the project that you want to create:
   a) In the **Project name** field, enter a project name.
   b) By default, the created project is located in the workspace current in use. If you do not want to use the default location for the project, clear the **Use default location** check box and click **Browse** to select a new location.
   c) Use the default version of the application module, or enter a new version in the **Version** field.
   d) Keep the **Create empty process** and **Create Application** check boxes selected to automatically create an empty process and an application when creating the project.
   e) Select the **Use Java configuration** check box if you want to create a Java module.

      A Java module provides the Java tooling capabilities.
   f) Click **Finish** to create the project.

Result

The project with the specified settings is displayed in the Project Explorer view.



Creating a Load SWIFT MT Schema Shared Resource

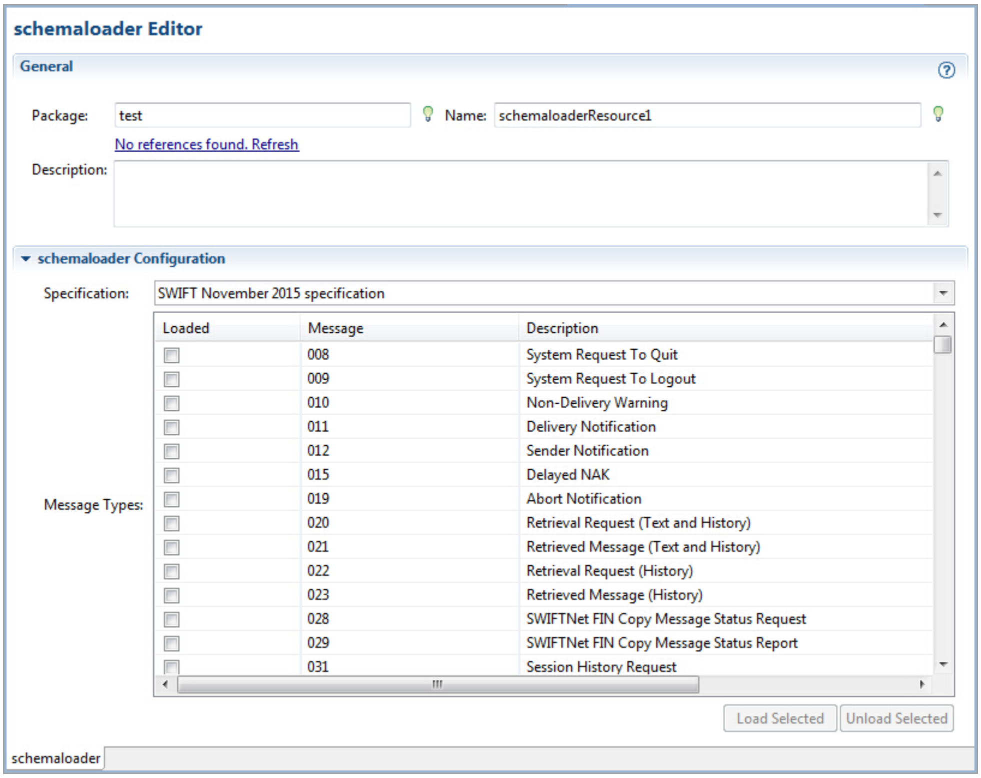
After creating a project, you can add a Load SWIFT MT Schema shared resource to select a SWIFT specification and message type. This shared resource is used for the Parse SWIFT MT, Render SWIFT MT, and Route SWIFT MT activities.

Prerequisites

The Load SWIFT MT Schema shared resource is available at the **Resources** level. Ensure that you have created a project, as described in Creating a Project.

Procedure

1. Expand the created project in the Project Explorer view.
2. Right-click the **Resources** folder and click **New** > **Load SWIFT MT Schema** to open the schemaloader wizard.
3. The resource folder, package name, and resource name of the Load SWIFT MT Schema are provided by default. If you do not want to use the default configurations, change them accordingly. Click **Finish** to open the schemaloader Editor.



schemaloader Editor

General

Package: test

Name: schemaloaderResource1

No references found. Refresh

Description:

schemaloader Configuration

Specification: SWIFT November 2015 specification

Message Types:

Loaded	Message	Description
□	008	System Request To Quit
□	009	System Request To Logout
□	010	Non-Delivery Warning
□	011	Delivery Notification
□	012	Sender Notification
□	015	Delayed NAK
□	019	Abort Notification
□	020	Retrieval Request (Text and History)
□	021	Retrieved Message (Text and History)
□	022	Retrieval Request (History)
□	023	Retrieved Message (History)
□	028	SWIFTNet FIN Copy Message Status Request
□	029	SWIFTNet FIN Copy Message Status Report
□	031	Session History Request

4. From the **Specification** list, select a SWIFT specification.

5. In the **Message Types** field, find the appropriate message type schema and select the check box in the Loaded column.

6. Click **Load Selected** to load the message type schemas.

After selecting or unselecting the message type, you have to click **Load Selected** or **Unload Selected** to make the message types available in the project.

Configuring a Process

After creating a project, an empty process is created. You can add activities to the empty process to complete a task. For example, parse a SWIFT MT message.

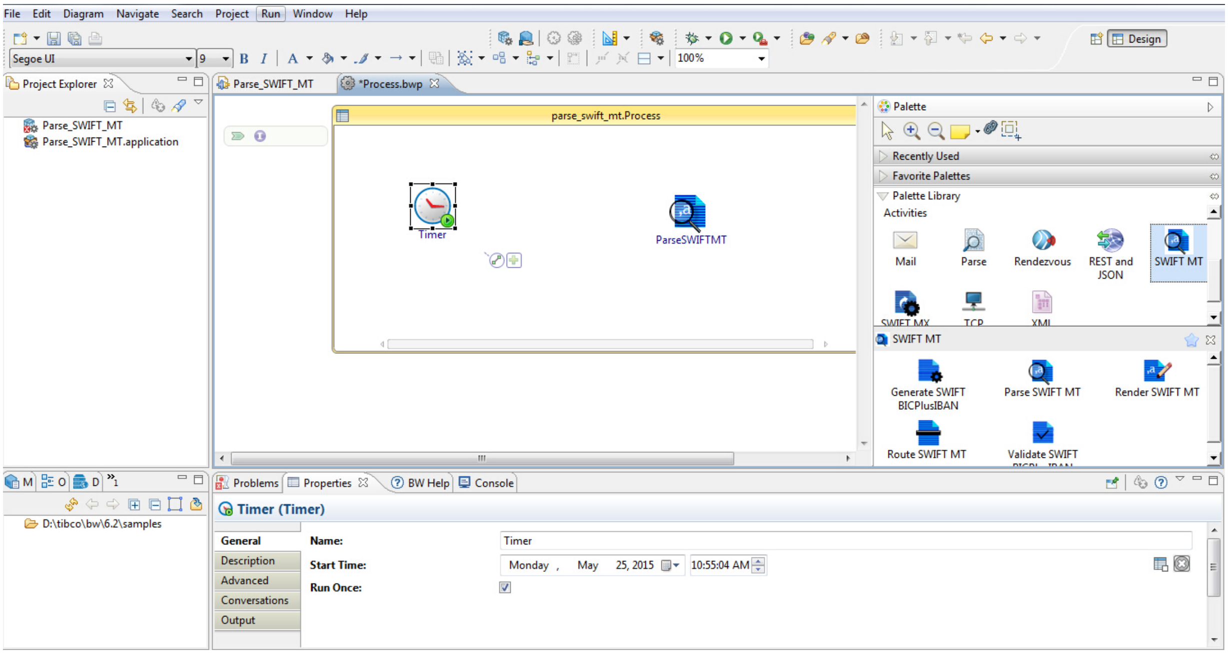
Prerequisites

Ensure that you have created an empty process when creating a project, see Creating a Project for more instructions.

Procedure

1. In the Project Explorer view, click the created project and open the empty process from the **Processes** folder.

2. Select an activity from the Palette view and drop it in the Process editor.
   For example, select and drop the Timer activity from the General Activities palette and the Parse SWIFT MT activity from the SWIFT MT palette.



Timer (Timer)

Name: Timer

Start Time: Monday , May 25, 2015 10:55:04 AM

Run Once: ☑

3. Drag the icon to create a transition between the added activities.
4. Configure the added Parse SWIFT MT activities, as described in SWIFT MT Palette.

   The Load SWIFT MT shared resource is required when configuring the SWIFT MT activities. See Creating a Load SWIFT MT Schema Shared Resource for more details.

5. Click **File** > **Save** to save the project.

Testing a Process

After configuring a process, you can test the process to check if the process completes your task.

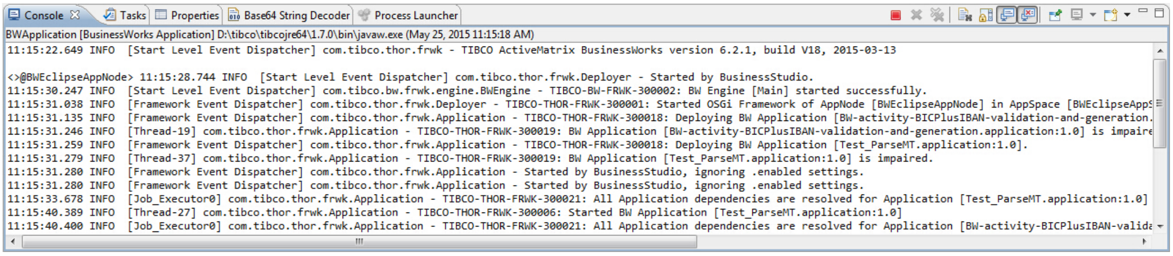
Prerequisites

Ensure that you have configured a process, as described in Configuring a Process .

Procedure

1. On the toolbar, click **Debug** > **Debug Configurations**.
2. Click **BusinessWorks Application** > **BWApplication** in the left panel.

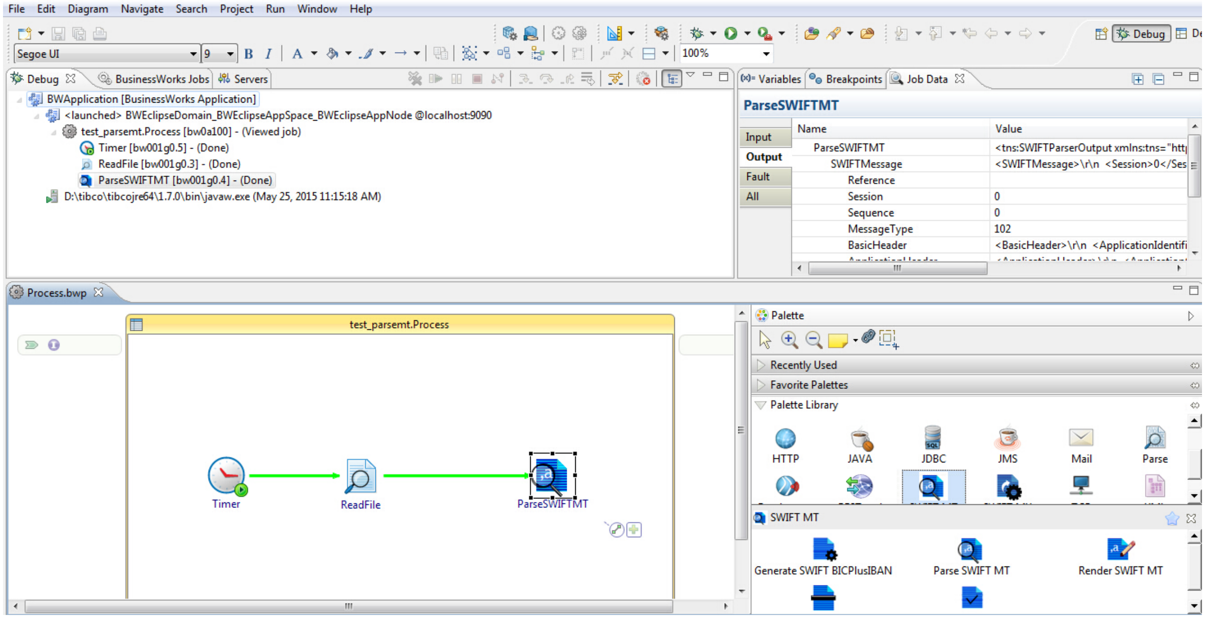
   By default, all the applications in the current workspace are selected in the **Applications** tab. Ensure that only the application you want to debug is selected in the **Applications** tab in the right panel.
3. Click **Debug** to test the process in the selected application.
   TIBCO Business Studio changes to the Debug perspective. The debug information is displayed in the Console view.



```
BWApplication [BusinessWorks Application] D:\tibco\tibcojre64\1.7.0\bin\javaw.exe (May 25, 2015 11:15:18 AM)
11:15:22.649 INFO  [Start Level Event Dispatcher] com.tibco.thor.frwk - TIBCO ActiveMatrix BusinessWorks version 6.2.1, build V18, 2015-03-13

<>@BWEclipseAppNode> 11:15:28.744 INFO  [Start Level Event Dispatcher] com.tibco.thor.frwk.Deployer - Started by BusinessStudio.
11:15:30.247 INFO  [Start Level Event Dispatcher] com.tibco.bw.frwk.engine.BWEngine - TIBCO-BW-FRWK-300002: BW Engine [Main] started successfully.
11:15:31.038 INFO  [Framework Event Dispatcher] com.tibco.thor.frwk.Deployer - TIBCO-THOR-FRWK-300001: Started OSGi Framework of AppNode [BWEclipseAppNode] in AppSpace [BWEclipseAppS
11:15:31.135 INFO  [Framework Event Dispatcher] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300018: Deploying BW Application [BW-activity-BICPlusIBAN-validation-and-generation.
11:15:31.246 INFO  [Thread-19] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300019: BW Application [BW-activity-BICPlusIBAN-validation-and-generation.application:1.0] is impaire
11:15:31.259 INFO  [Framework Event Dispatcher] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300018: Deploying BW Application [Test_ParseMT.application:1.0].
11:15:31.279 INFO  [Thread-37] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300019: BW Application [Test_ParseMT.application:1.0] is impaired.
11:15:31.280 INFO  [Framework Event Dispatcher] com.tibco.thor.frwk.Application - Started by BusinessStudio, ignoring .enabled settings.
11:15:31.280 INFO  [Framework Event Dispatcher] com.tibco.thor.frwk.Application - Started by BusinessStudio, ignoring .enabled settings.
11:15:33.678 INFO  [Job_Executor0] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300021: All Application dependencies are resolved for Application [Test_ParseMT.application:1.0]
11:15:40.389 INFO  [Thread-27] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300006: Started BW Application [Test_ParseMT.application:1.0]
11:15:40.400 INFO  [Job_Executor0] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300021: All Application dependencies are resolved for Application [BW-activity-BICPlusIBAN-valida
```

4. In the **Debug** tab, expand the running process and click an activity.
5. In the upper-right corner, click the **Job Data** tab, and then click the **Output** tab to check the activity output.



```
BWApplication [BusinessWorks Application]
  <launched> BWEclipseDomain_BWEclipseAppSpace_BWEclipseAppNode @localhost:9090
    test_parsemt.Process [bw0a100] - (Viewed job)
      Timer [bw001g0.5] - (Done)
      ReadFile [bw001g0.3] - (Done)
      ParseSWIFTMT [bw001g0.4] - (Done)
  D:\tibco\tibcojre64\1.7.0\bin\javaw.exe (May 25, 2015 11:15:18 AM)
```

ParseSWIFTMT

Name	Value
ParseSWIFTMT	<tns:SWIFTParserOutput xmlns:tns="htt
SWIFTMessage	<SWIFTMessage>\r\n <Session>0</Ses
Reference	
Session	0
Sequence	0
MessageType	102
BasicHeader	<BasicHeader>\r\n <ApplicationIdentifi

Deploying a Process

After testing, if the configured process works as expected, you can deploy the application that contains the configured process into a runtime environment, and then use the **`bwadmin`** utility to manage the deployed application.

Before deploying an application, you must generate an application archive, which is an enterprise archive (EAR) file that is created in TIBCO Business Studio.

Deploying an application involves the following tasks:

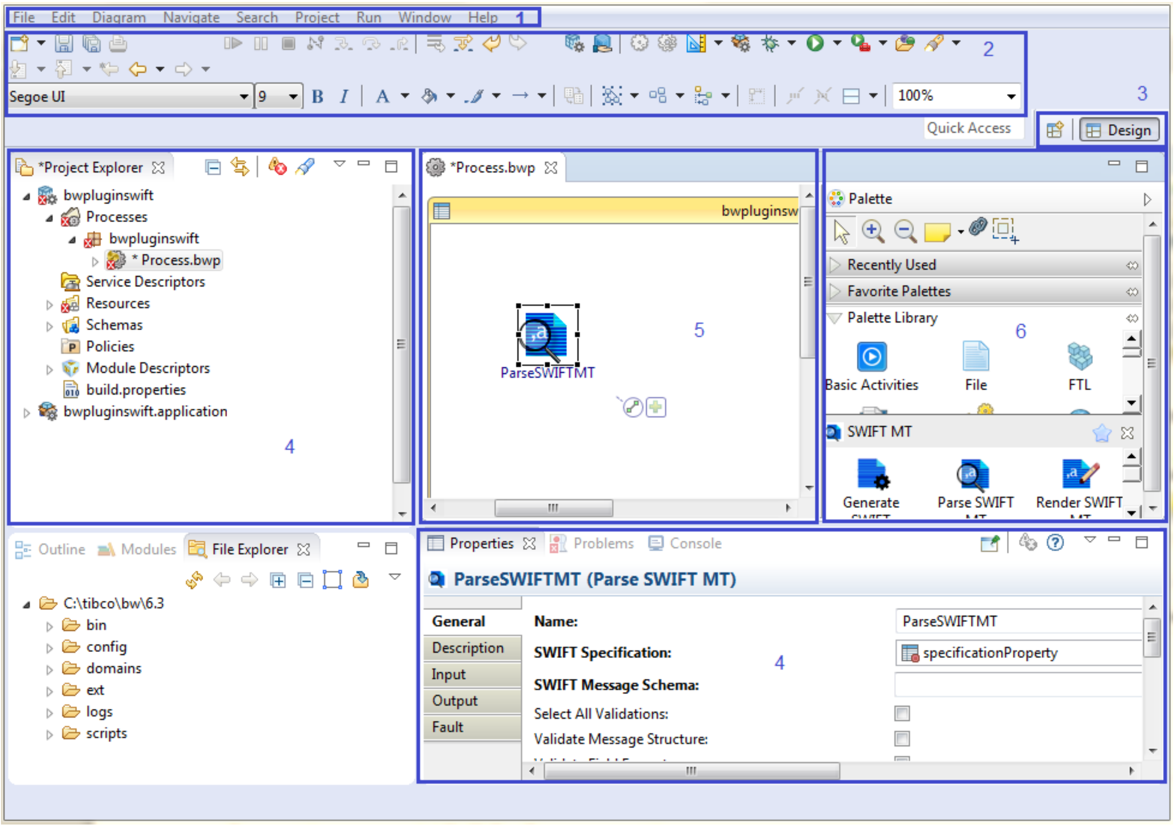
1. Uploading an application archive.
2. Deploying an application archive.
3. Starting an application.

See *TIBCO ActiveMatrix BusinessWorks Administration* for more details about how to deploy an application.

TIBCO Business Studio Overview

TIBCO Business Studio is an Eclipse-based integration development environment that is used to design, develop, and test ActiveMatrix BusinessWorks applications.

TIBCO Business Studio provides a workbench in which you can create, manage, and navigate resources in your workspace. A *workspace* is the central location on your machine where all data files are stored.



Name: ParseSWIFTMT

SWIFT Specification: specificationProperty

SWIFT Message Schema:

Select All Validations: □

Validate Message Structure: □

The workbench consists of:

- **Menu**: contains menu items such as File, Edit, Diagram, Navigate, Search, Project, Run, Window, and Help.
- **Toolbar**: contains buttons for frequently used commands such as New, Save, Enable/Disable Business Studio Capabilities, Create a new BusinessWorks Application Module, Create a new BusinessWorks Shared Module, Debug, Run, and so on.
- **Perspective**: contains an initial set and layout of views that are required to perform a certain task. TIBCO Business Studio launches the Modeling perspective by default. You can change the perspective from the menu **Window** > **Open Perspective** > ***Perspective_Name***.
- **View**: displays resources. For example, the Project Explorer view displays the ActiveMatrix BusinessWorks applications, modules, and other resources in your workspace, and the Properties view displays the properties for the selected resource. You can open a view from the menu **Window** > **Show View** > ***View_Name***.
- **Editor**: provides a canvas to configure, edit, or browse a resource. Double-click a resource in a view to open the appropriate editor for the selected resource. For example, double-click an ActiveMatrix BusinessWorks process `MortgageAppConsumer.bwp` in the Project Explorer view to open the process in the editor.
- **Palette**: contains a set of widgets and a palette library. A *palette* groups activities that perform similar tasks, and provides quick access to activities when configuring a process.

Creating a Load SWIFT MX Schema Shared Resource

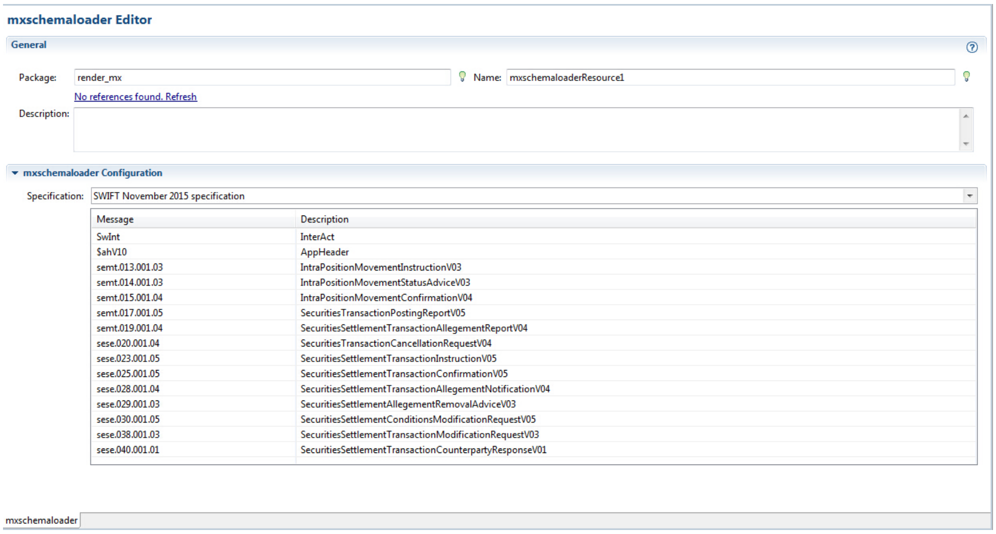
You can use the Load SWIFT MX Schema shared resource to load the SWIFT MX schema.

Prerequisites

Before creating the Load SWIFT MX Schema shared resource, you have to download the required XSD files from https://www2.swift.com/uhbonline/books/a2z/standards_mx.htm, and save them to `TIBCO_HOME/bw/palettes/swift/version_number/bin/xsd/specification_year` directory.

Procedure

1. Expand the created project in the Project Explorer view.
2. Right-click the **Resources** folder and click **New** > **Load SWIFT MX Schema** to open the mxschemaloader wizard.
3. The resource folder, package name, and resource name are provided by default. If you do not want to use the default configurations, change them accordingly. Click **Finish** to open the mxschemaloader Editor.
   All the available SWIFT MX schemas are displayed in the dialog, and you can use this shared resource in your project.



mxschemaloader Editor

General

Package: render_mx

Name: mxschemaloaderResource1

No references found. Refresh

Description:

mxschemaloader Configuration

Specification: SWIFT November 2015 specification

Message	Description
SwInt	InterAct
$ahV10	AppHeader
semt.013.001.03	IntraPositionMovementInstructionV03
semt.014.001.03	IntraPositionMovementStatusAdviceV03
semt.015.001.04	IntraPositionMovementConfirmationV04
semt.017.001.05	SecuritiesTransactionPostingReportV05
semt.019.001.04	SecuritiesSettlementTransactionAllegementReportV04
sese.020.001.04	SecuritiesTransactionCancellationRequestV04
sese.023.001.05	SecuritiesSettlementTransactionInstructionV05
sese.025.001.05	SecuritiesSettlementTransactionConfirmationV05
sese.028.001.04	SecuritiesSettlementTransactionAllegementNotificationV04
sese.029.001.03	SecuritiesSettlementAllegementRemovalAdviceV03
sese.030.001.05	SecuritiesSettlementConditionsModificationRequestV05
sese.038.001.03	SecuritiesSettlementTransactionModificationRequestV03
sese.040.001.01	SecuritiesSettlementTransactionCounterpartyResponseV01

> If no XSD file is matched for the message type in the Message column, the message type is unable to use.

4. Save this shared resource.

Load SWIFT MT Schema Shared Resource

You can use the Load SWIFT MT Schema shared resource to select a SWIFT Specification and load the message type schemas. The schemas loaded using the Load SWIFT MT shared resource are used by the Parse SWIFT MT, Route SWIFT MT, and Render SWIFT MT activities. You can add only one Load MT Schema shared resource in a project.

General

The following table lists the configurations in the **General** panel of the Load SWIFT MT Schema shared resource:

Field	Module Property?	Description
Package	No	The name of the package where the new shared resource is added.
Name	No	The name to be displayed as the label for the shared resource in the process.
Description	No	A short description for this shared resource.

schemaloader Configuration

The following table lists the configurations in the **schemaloader Configuration** panel of the Load SWIFT MT Schema shared resource:

Field	Description
Specification	You can select a SWIFT specification from the **Specification** list. The default specification is `SWIFT November 2015 specification`.
Message Types	All the message types supported by the selected SWIFT specification. Select the multiple check boxes next to the message type and click **Load Selected** to load message schemas to a project; click **Unload Selected** to unload the message schemas. If you do not select all the schemas at once, when another schema is initiated, the previous selection might be removed from the project when other schema is initiated.

See Creating a Load SWIFT MT Schema Shared Resource to create the Load SWIFT MT Schema shared resource.

Load SWIFT MX Schema Shared Resource

The Load SWIFT MX Schema shared resource is used to select a SWIFT Specification and load or unload the MX message schemas. The Parse SWIFT MX and Render SWIFT MX activities use the schemas that are loaded by the Load SWIFT MX shared resource. You can add only one Load MX Schema shared resource in a project.

General

The following table lists the configurations in the **General** panel of the Load SWIFT MX Schema shared resource:

Field	Module Property?	Description
Package	No	The name of the package where the new shared resource is added.
Name	No	The name to be displayed as the label for the shared resource in the process.
Description	No	A short description for this shared resource.

Supported MX messages

The plug-in supports part of SWIFT MX Standard Release 2015, and therefore, supports a certain set of MX messages that conform to SWIFT MX Standard Release 2015.

The plug-in supports extended rules validation for the following MX messages:

```
semt.013.001.03
semt.014.001.03
semt.015.001.04
semt.017.001.05
semt.018.001.06
semt.019.001.04
semt.021.001.05
semt.023.001.01
sese.020.001.04
sese.023.001.05
sese.024.001.06
sese.025.001.05
sese.027.001.04
sese.028.001.04
sese.029.001.03
sese.030.001.05
sese.031.001.06
sese.038.001.03
sese.039.001.03
sese.040.001.01
setr.027.001.03
setr.029.001.01
setr.030.001.01
setr.044.001.02

acmt.001.001.05
acmt.002.001.05
acmt.003.001.05
acmt.004.001.03
acmt.005.001.03
acmt.006.001.04
camt.040.001.04
camt.041.001.04
camt.042.001.04
camt.043.001.04
camt.044.001.03
camt.045.001.03
reda.001.001.04
reda.002.001.04
reda.004.001.02
reda.005.001.02
semt.001.001.03
semt.002.001.02
semt.003.001.02
semt.004.001.02
semt.005.001.02
semt.006.001.02
semt.007.001.02
semt.012.001.01
sese.001.001.06
sese.002.001.06
sese.003.001.06
sese.004.001.06
sese.005.001.06
sese.006.001.06
sese.007.001.06
sese.008.001.06
sese.009.001.04
sese.010.001.04
sese.011.001.04
sese.012.001.06
sese.013.001.06
sese.014.001.06
sese.018.001.04
sese.019.001.03
setr.001.001.03
setr.002.001.03
setr.003.001.03
setr.004.001.03
setr.005.001.03
setr.006.001.03
setr.007.001.03
setr.008.001.03
setr.009.001.03
setr.010.001.03
setr.011.001.03
setr.012.001.03
setr.013.001.03
setr.014.001.03
setr.015.001.03
setr.016.001.03
setr.017.001.03
setr.018.001.03
setr.047.001.01
setr.048.001.01
setr.049.001.01
setr.050.001.01
setr.051.001.01
setr.052.001.01
setr.053.001.01
setr.054.001.01
setr.055.001.01
setr.056.001.01
setr.057.001.01
setr.058.001.01
setr.059.001.01
setr.060.001.01
setr.061.001.01
setr.062.001.01
setr.064.001.01
setr.065.001.01
setr.066.001.01

camt.029.001.05
camt.052.001.05
camt.053.001.05
camt.056.001.04
pacs.002.001.06
pacs.003.001.05
pacs.004.001.05
pacs.007.001.05
pacs.008.001.05
```

SWIFT MT Palette

The SWIFT MT palette contains the following five activities:

- Parse SWIFT MT Activity
- Render SWIFT MT Activity
- Route SWIFT MT Activity
- Generate SWIFT BICPlusIBAN Activity
- Validate SWIFT BICPlusIBAN Activity

Parse SWIFT MT Activity

You can use the Parse SWIFT MT activity to validate the MT message and parse it to XML format.

General

In the **General** tab, you can specify a message schema and select multiple validation types to parse and validate an MT message.

The following table lists the configurations in the **General** tab of the Parse SWIFT MT activity:

Field	Module Property?	Description
Name	No	Specify the name to be displayed as the label for the activity in the process.
SWIFT Specification	Yes	Click 🔍 to select a Load SWIFT MT Schema shared resource. If no matching Load SWIFT MT Schema shared resource is found, click **Create Shared Resource** to create one. For more details, see Creating a Load SWIFT MT Schema Shared Resource.
SWIFT Message Schema	No	Click 🔍 to select a message type schema which you want to configure for the activity.
Select All Validations	No	Select this check box to enable all the validations.
Validate Message Structure	No	Select this check box to validate the SWIFT message structure. See Guidelines for Validating the Message Structure for more information.

Field	Module Property?	Description
Validate Field Format	No	Select this check box to validate the SWIFT message format. This format specification includes character set membership, mandatory subfield presence, fixed and maximum length subfield enforcement, and multiple line subfield line count enforcement. All SWIFT text and header fields must follow their format specifications exactly. Certain field format errors are considered parse errors and they occur even when this validation feature is disabled. For example, the format of the field 20C of an MT 500 is `4!c//16x`, which means the string must be as follows: • 4 alphabetic characters fixed length • double forward slash (//) • not more than 16 characters which belongs to the X S.W.I.F.T. character set
Validate Data Types	No	Select this check box to validate the SWIFT message data type. SWIFT header and text subfields, such as message types, currency codes, amounts, data, and times are all validated for correct format. The syntactic BIC and LTA validations are part of this feature. Validation against the BIC database requires the use of the **Validate BIC/BEI** field.
Validate Qualifier Code Words	No	Select this check box to validate the qualifier code word. SWIFT code words are various qualifier-dependent values used in various text subfields. The SWIFT network is currently in a transition for code word validation, and not all customers might want to validate code words.
Validate BIC/BEI	No	Select this check box to validate the SWIFT message BIC/BEI codes against a BIC directory. SWIFT BIC and BEI codes are entity identifiers that are displayed in various subfields of the text block. Validating them takes a lot of time and memory, because data must be searched in a very large database. At the initialization, an `FI.dat` BIC directory is provided to the validator, and then full BIC/BEI validation is performed against the provided directory. If no BIC directory is provided, the plug-in uses the default BIC file. • Use the latest BIC file provided by SWIFT for up-to-date BIC or BEI validation. • An FI.dat BIC directory is provided by the plugin as the default BIC file, but you can also provide the most recent BIC file from Swift to overwrite the default file in validation.

Field	Module Property?	Description
Validate Field-Level Semantics	No	Select this check box to validate the SWIFT text blocks field-level semantic. Field-level rules are SWIFT network validation rules that relate data in the subfields of a single field. Examples include `"field 22H must have one of subfield Narrative, or Amount, but not both"`. This validation is not performed for SWIFT-to-user system messages even if this feature is enabled. This situation happens because these messages are always from SWIFT, and they are assumed to be valid.
Validate Message-Level Semantics	No	Select this check box to enable message-level semantic validation of SWIFT text blocks. Message-level rules are SWIFT network validation rules that relate data in different parts of a message. Examples include `"if field 22H in sequence A has value X, then field 98A in sequence B must be present"`. These rules are complex and slow the validation process. Furthermore, customers frequently enforce these rules downstream of the validator in their own business logic, so it is not always necessary to enable this sort of validation. This validation is not performed for SWIFT-to-user system messages even if this feature is enabled. This situation happens because these messages are always from SWIFT, and they are assumed to be valid.
Validate Structured Narrative	No	Select this check box to validate the structure narrative of SWIFT message headers. Various SWIFT narrative tags, for example, 71B and 72 are required structures in various messages. The structure typically consists of keywords, continuation characters, or both, occasionally in a particular order.
Validate Advanced Header	No	Select this check box to enable advanced validation of SWIFT message headers. Such validation includes: • Enforcing the order of tags in the user header section • Enforcing the order of tags in the trailer section • Enforcing relationships between delivery options, message types, and obsolescence periods in the application header
BIC Code File(FI.dat)	Yes	Click 🔍 and navigate to the location of the BIC or BICPlus file. Select the file and click **Open** to load it. • The BIC or BICPlus file is extremely large and, if specified, causes the increase of plug-in startup time. If this file is not specified, the default `FI.dat` file is used. • An FI.dat BIC directory is provided by the plugin as the default BIC file, but you can also provide the most recent BIC file from Swift to overwrite the default file in validation.

Field	Module Property?	Description
ISO3166 Country Code File(CT.dat)	Yes	Click 🔍 to navigate to the location of the country code data file. Select the file and click **Open** to load it. The ISO3166 country codes are used if the file is specified here. Otherwise, an internal database of country codes is used.
ISO4217 Currency Code File(CU.dat)	Yes	Click 🔍 to navigate to the location of the currency code data file. Select the file and click **Open** to load it. The ISO4217 currency codes are used if the file is specified here. Otherwise an internal database of currency codes is used.
Validation Filter File(ValidationFilter.xml)	Yes	Click 🔍 to navigate to the location of the `ValidationFilter.xml` file. Select the file and click **Open** to load it. See Using Validation Filters for more information about the validation filters feature.

Description

In the **Description** tab, you can enter a short description for the Parse SWIFT MT activity.

Input

The following table lists the input element in the **Input** tab of the Parse SWIFT MT activity:

Input Item	Data Type	Description
`FINMessage`	String	MT message in String format.

Output

In the **Output** tab, you can find the parse results.

The following table lists the output element in the **Output** tab of the Parse SWIFT MT activity:

Output Item	Data Type	Description
`SWIFTMessage`	XML	XML representation of the input MT message. The contents of the schema are determined by the SWIFT Message schema specified in the **General** tab.

Fault

In the **Fault** tab, you can find the error code and error message of the Parse SWIFT MT activity. See Error Codes for more detailed explanation of errors.

The following table lists the error schema elements in the **Fault** tab of the Parse SWIFT MT activity:

Error Schema Element	Data Type	Description
`ValidationException`	String	The MT messages have validation errors when the validation does not follow the selected SWIFT specification.
`SwiftException`	String	The activity cannot locate the metadata directory or it fails to initialize the metadata.

Render SWIFT MT Activity

You can use the Render SWIFT MT activity to render an MT message from XML format to the MT message format.

General

In the **General** tab, you can specify a message schema and select multiple validation types to render and validate the MT message in XML format.

The following table lists the configurations in the **General** tab of the Render SWIFT MT activity:

Field	Module Property?	Description
Name	No	Specify the name to be displayed as the label for the activity in the process.
SWIFT Specification	Yes	Click 🔍 to select a Load SWIFT MT Schema shared resource. If no matching Load SWIFT MT Schema shared resource is found, click **Create Shared Resource** to create one. For more details, see Creating a Load SWIFT MT Schema Shared Resource.
SWIFT Message Schema	No	Click 🔍 to select a message type schema which you want to configure for the activity.
Select All Validations	No	Select this check box to enable all the validations.
Validate Message Structure	No	Select this check box to validate the SWIFT message structure. See Guidelines for Validating the Message Structure for more information.

Field	Module Property?	Description
Validate Field Format	No	Select this check box to validate the SWIFT message format. This format specification includes character set membership, mandatory subfield presence, fixed and maximum length subfield enforcement, and multiple line subfield line count enforcement. All SWIFT text and header fields must follow their format specifications exactly. Certain field format errors are considered parse errors and they occur even when this validation feature is disabled.
Validate Data Types	No	Select this check box to validate the SWIFT message data type. SWIFT header and text subfields, such as message types, currency codes, amounts, data, and times are all validated for correct format. The syntactic BIC and LTA validations are part of this feature. Validation against the BIC database requires the use of the **Validate BIC/BEI** field.
Validate Qualifier Code Words	No	Select this check box to validate the qualifier code word. SWIFT code words are various qualifier-dependent values used in various text subfields. The SWIFT network is currently in a transition for code word validation, and not all customers might want to validate code words.
Validate BIC/BEI	No	Select this check box to validate the SWIFT message BIC/BEI codes against a BIC directory. SWIFT BIC and BEI codes are entity identifiers that are displayed in various subfields of the text block. Validating them takes a lot of time and memory, because data must be searched in a very large database. At the initialization, an `FI.dat` BIC directory is provided to the validator, and then full BIC/BEI validation is performed against the provided directory. If no BIC directory is provided, the plug-in uses the default BIC file. Use the latest BIC file provided by SWIFT for up-to-date BIC or BEI validation.
Validate Field-Level Semantics	No	Select this check box to validate the SWIFT text blocks field-level semantic. Field-level rules are SWIFT network validation rules that relate data in the subfields of a single field. Examples include `"field 22H must have one of subfield Narrative, or Amount, but not both"`. This validation is not performed for SWIFT-to-user system messages even if this feature is enabled. This situation happens because these messages are always from SWIFT, and they are assumed to be valid.

Field	Module Property?	Description
Validate Message-Level Semantics	No	Select this check box to enable message-level semantic validation of SWIFT text blocks. Message-level rules are SWIFT network validation rules that relate data in different parts of a message. Examples include `"if field 22H in sequence A has value X, then field 98A in sequence B must be present"`. These rules are complex and slow the validation process. Furthermore, customers frequently enforce these rules downstream of the validator in their own business logic, so it is not always necessary to enable this sort of validation. This validation is not performed for SWIFT-to-user system messages even if this feature is enabled. This situation happens because these messages are always from SWIFT, and they are assumed to be valid.
Validate Structured Narrative	No	Select this check box to validate the structure narrative of SWIFT message headers. Various SWIFT narrative tags, for example, 71B and 72 are required structures in various messages. The structure typically consists of keywords, continuation characters, or both, occasionally in a particular order.
Validate Advanced Header	No	Select this check box to enable advanced validation of SWIFT message headers. Such validation includes: • Enforcing the order of tags in the user header section • Enforcing the order of tags in the trailer section • Enforcing relationships between delivery options, message types, and obsolescence periods in the application header
BIC Code File(FI.dat)	Yes	Click 🔍 and navigate to the location of the BIC or BICPlus file. Select the file and click **Open** to load it. The BIC or BICPlus file is extremely large and, if specified, causes the increase of plug-in startup time. If this file is not specified, the default `FI.dat` file is used.
ISO3166 Country Code File(CT.dat)	Yes	Click 🔍 to navigate to the location of the country code data file. Select the file and click **Open** to load it. The ISO3166 country codes are used if the file is specified here. Otherwise, an internal database of country codes is used.

Field	Module Property?	Description
ISO4217 Currency Code File(CU.dat)	Yes	Click 🔍 to navigate to the location of the currency code data file. Select the file and click **Open** to load it. The ISO4217 currency codes are used if the file is specified here. Otherwise an internal database of currency codes is used.
Validation Filter File(ValidationFilter.xml)	Yes	Click 🔍 to navigate to the location of the `ValidationFilter.xml` file. Select the file and click **Open** to load it. See Using Validation Filters for more information about the validation filters feature.

Description

In the **Description** tab, you can enter a short description for the Render SWIFT MT activity.

Input

The following table lists the input element in the **Input** tab of the Render SWIFT MT activity:

Input Item	Data Type	Description
`SWIFTMessage`	XML	XML representation of the input MT message. The contents of the schema are determined by the SWIFT Message schema specified in the **General** tab.

Output

In the **Output** tab, you can find the render results.

The following table lists the output element in the **Output** tab of the Render SWIFT MT activity:

Output Item	Data Type	Description
`FINMessage`	String	MT message in String format.

Fault

In the **Fault** tab, you can find the error code and error message of the Render SWIFT MT activity. See Error Codes for more detailed explanation of the error.

The following table lists the error schema elements in the **Fault** tab of the Render SWIFT MT activity:

Error Schema Element	Data Type	Description
`ValidationException`	String	The MT messages have validation errors when the validation does not follow the selected SWIFT specification.

Error Schema Element	Data Type	Description
`SwiftException`	String	The activity cannot locate the metadata directory or it fails to initialize the metadata.

Route SWIFT MT Activity

You can use the Route SWIFT MT activity to route messages to different activities based on the message types.

General

In the **General** tab, you can specify a message schema and select multiple validation types to route and validate an MT message.

The following table lists the configurations in the **General** tab of the Route SWIFT MT activity:

Field	Module Property?	Description
Name	No	Specify the name to be displayed as the label for the activity in the process.
SWIFT Specification	Yes	Click 🔍 to select a Load SWIFT MT Schema shared resource. If no matching Load SWIFT MT Schema shared resource is found, click **Create Shared Resource** to create one. For more details, see Creating a Load SWIFT MT Schema Shared Resource.
Select All Validations	No	Select this check box to enable all the validations.
Validate Message Structure	No	Select this check box to validate the SWIFT message structure. See Guidelines for Validating the Message Structure for more information.
Validate Field Format	No	Select this check box to validate the SWIFT message format. This format specification includes character set membership, mandatory subfield presence, fixed and maximum length subfield enforcement, and multiple line subfield line count enforcement. All SWIFT text and header fields must follow their format specifications exactly. Certain field format errors are considered parse errors and they occur even when this validation feature is disabled.
Validate Data Types	No	Select this check box to validate the SWIFT message data type. SWIFT header and text subfields, such as message types, currency codes, amounts, data, and times are all validated for correct format. The syntactic BIC and LTA validations are part of this feature. Validation against the BIC database requires the use of the **Validate BIC/BEI** field.

Field	Module Property?	Description
Validate Qualifier Code Words	No	Select this check box to validate the qualifier code word. SWIFT code words are various qualifier-dependent values used in various text subfields. The SWIFT network is currently in a transition for code word validation, and not all customers might want to validate code words.
Validate BIC/BEI	No	Select this check box to validate the SWIFT message BIC/BEI codes against a BIC directory. SWIFT BIC and BEI codes are entity identifiers that are displayed in various subfields of the text block. Validating them takes a lot of time and memory, because data must be searched in a very large database. At the initialization, an `FI.dat` BIC directory is provided to the validator, and then full BIC/BEI validation is performed against the provided directory. If no BIC directory is provided, the plug-in uses the default BIC file. Use the latest BIC file provided by SWIFT for up-to-date BIC or BEI validation.
Validate Field-Level Semantics	No	Select this check box to validate the SWIFT text blocks field-level semantic. Field-level rules are SWIFT network validation rules that relate data in the subfields of a single field. Examples include `"field 22H must have one of subfield Narrative, or Amount, but not both"`. This validation is not performed for SWIFT-to-user system messages even if this feature is enabled. This situation happens because these messages are always from SWIFT, and they are assumed to be valid.
Validate Message-Level Semantics	No	Select this check box to enable message-level semantic validation of SWIFT text blocks. Message-level rules are SWIFT network validation rules that relate data in different parts of a message. Examples include `"if field 22H in sequence A has value X, then field 98A in sequence B must be present"`. These rules are complex and slow the validation process. Furthermore, customers frequently enforce these rules downstream of the validator in their own business logic, so it is not always necessary to enable this sort of validation. This validation is not performed for SWIFT-to-user system messages even if this feature is enabled. This situation happens because these messages are always from SWIFT, and they are assumed to be valid.
Validate Structured Narrative	No	Select this check box to validate the structure narrative of SWIFT message headers. Various SWIFT narrative tags, for example, 71B and 72 are required structures in various messages. The structure typically consists of keywords, continuation characters, or both, occasionally in a particular order.

Field	Module Property?	Description
Validate Advanced Header	No	Select this check box to enable advanced validation of SWIFT message headers. Such validation includes: • Enforcing the order of tags in the user header section • Enforcing the order of tags in the trailer section • Enforcing relationships between delivery options, message types, and obsolescence periods in the application header
BIC Code File(FI.dat)	Yes	Click 🔍 and navigate to the location of the BIC or BICPlus file. Select the file and click **Open** to load it. The BIC or BICPlus file is extremely large and, if specified, causes the increase of plug-in startup time. If this file is not specified, the default `FI.dat` file is used.
ISO3166 Country Code File(CT.dat)	Yes	Click 🔍 to navigate to the location of the country code data file. Select the file and click **Open** to load it. The ISO3166 country codes are used if the file is specified here. Otherwise, an internal database of country codes is used.
ISO4217 Currency Code File(CU.dat)	Yes	Click 🔍 to navigate to the location of the currency code data file. Select the file and click **Open** to load it. The ISO4217 currency codes are used if the file is specified here. Otherwise an internal database of currency codes is used.
Validation Filter File(ValidationFilter.xml)	Yes	Click 🔍 to navigate to the location of the `ValidationFilter.xml` file. Select the file and click **Open** to load it. See Using Validation Filters for more information about the validation filters feature.

Description

In the **Description** tab, you can enter a short description for the Route SWIFT MT activity.

Input

The following table lists the input element in the **Input** tab of the Route SWIFT MT activity:

Input Item	Data Type	Description
`FINMessage`	String	MT message in String format.

Output

In the **Output** tab, you can find the route results.

The following table lists the output elements in the **Output** tab of the Route SWIFT MT activity:

Output Item	Data Type	Description
`MessageType`	String	Message type in String format.
`FINMessage`	String	MT message in String format.

Fault

In the **Fault** tab, you can find the error code and error message of the Route SWIFT MT activity. See Error Codes for more detailed explanation of the error.

The following table lists the error schema elements in the **Fault** tab of the Route SWIFT MT activity:

Error Schema Element	Data Type	Description
`ValidationException`	String	The MT messages have validation errors when the validation does not follow the selected SWIFT specification.
`SwiftException`	String	The activity cannot locate the metadata directory or it fails to initialize the metadata.

Generate SWIFT BICPlusIBAN Activity

You can use the Generate SWIFT BICPlusIBAN activity to generate an IBAN.

General

In the **General** tab, you can specify the BANK Directory and IBAN Plus data file to generate an IBAN.

The following table lists the configurations in the **General** tab of the Generate SWIFT BICPlusIBAN activity:

Field	Module Property?	Description
Name	No	Specify the name to be displayed as the label for the activity in the process.

Field	Module Property?	Description
BANK Directory and IBAN Plus	Yes	Click 🔍 and navigate to the directory where the BANK Directory and IBAN Plus data file is located. Select the file and click **OK** to load it. You have to download the required BANK Directory and IBAN Plus data file from the SWIFT official website, and ensure the following files are exist inside: • `IBANPLUS_V*_FULL_year****` file (IBAN information), required. • `IBANSTRUCTURE_FULL_year****` file (IBAN structure information), required. • `EXCLUSIONLIST_V*_FULL_year****` file (National IDs cannot be in IBANs), required. • `COUNTRY_CODE_year****` file (Country code information), required. • `BANKDIRECTORYPLUS_V*_FULL_year****` file (Bank information), required.

Description

In the **Description** tab, you can enter a short description for the Generate SWIFT BICPlusIBAN activity.

Input

The following table lists the input elements in the **Input** tab of the Generate SWIFT BICPlusIBAN activity:

Input Item	Data Type	Description
`CountryCode`	String	Specify the 2-character country code that is required to generate an IBAN.
`BBAN`	String	Specify the BBAN that is required to generate an IBAN.

Output

In the **Output** tab, you can find the generated results.

The following table lists the output element in the **Output** tab of the Generate SWIFT BICPlusIBAN activity:

Output Item	Data Type	Description
`IBAN`	String	IBAN in String format.

Fault

In the **Fault** tab, you can find the error code and error message of the Generate SWIFT BICPlusIBAN activity. See Error Codes for more detailed explanation of errors.

The following table lists the error schema elements in the **Fault** tab of the Generate SWIFT BICPlusIBAN activity:

Error Schema Element	Data Type	Description
`ValidationException`	String	The MT messages have validation errors when the validation does not follow the selected SWIFT specification.
`SwiftException`	String	The activity cannot locate the metadata directory or it fails to initialize the metadata.

Validate SWIFT BICPlusIBAN Activity

You can use the Validate SWIFT BICPlusIBAN activity to validate an IBAN.

General

In the **General** tab, you can specify the BANK Directory and IBAN Plus data file to validate an IBAN.

The following table lists the configurations in the **General** tab of the Validate SWIFT BICPlusIBAN activity:

Field	Module Property?	Description
Name	No	Specify the name to be displayed as the label for the activity in the process.
BANK Directory and IBAN Plus	Yes	Click 🔍 and navigate to the directory where the BANK Directory and IBAN Plus data file is located. Select the file and click **OK** to load it. You have to download the required BANK Directory and IBAN Plus data file from the SWIFT official website, and ensure the following files are exist inside: • `IBANPLUS_V*_FULL_year****` file (IBAN information), required. • `IBANSTRUCTURE_FULL_year****` file (IBAN structure information), required. • `EXCLUSIONLIST_V*_FULL_year****` file (National IDs cannot be in IBANs), required. • `COUNTRY_CODE_year****` file (Country code information), required. • `BANKDIRECTORYPLUS_V*_FULL_year****` file (Bank information), required.

Description

In the **Description** tab, you can enter a short description for the Validate SWIFT BICPlusIBAN activity.

Input

In the **Input** tab, you can specify the IBAN that you want to validate.

The following table lists the input elements in the **Input** tab of the Validate SWIFT BICPlusIBAN activity:

Input Item	Data Type	Description
IBAN	String	Specify the IBAN that you want to validate.
BIC	String	Specify the BIC that you want to validate.
BranchCode	String	Specify the BranchCode that you want to validate.

Output

In the **Output** tab, you can find the validation results.

The following table lists the output element in the **Output** tab of the Validate SWIFT BICPlusIBAN activity:

Output Item	Data Type	Description
BICPlusIBANRecord	XML	Displays the IBAN validation results.

Fault

In the **Fault** tab, you can find the error code and error message of the Search Entry activity. See Error Codes for more detailed explanation of errors.

The following table lists the error schema elements in the **Fault** tab of the Validate SWIFT BICPlusIBAN activity:

Error Schema Element	Data Type	Description
ValidationException	String	The MT messages have validation errors when the validation does not follow the selected SWIFT specification.
SwiftException	String	The activity cannot locate the metadata directory or it fails to initialize the metadata.

Guidelines for Validating the Message Structure

When validating an MT message structure, the following guidelines are used:

- The fields of all required basic and application headers have the correct length.
- The basic header and service identifiers are valid.
- Application input and output indicators are valid.

- Application input delivery monitoring and obsolescence cannot be displayed in the message unless the priority is displayed.
- Application input obsolescence cannot be displayed in the message unless the delivery monitoring is displayed.
- The user header is displayed if required.
- Message types requiring a user header field validation tag 119 have such a tag.
- No duplicate user header or trailer tags are found.
- No invalid user header, trailer, or system acknowledgment tags are found.
- The lengths of all user header, trailer, and system acknowledgment tag values are correct.
- Message types requiring user header field validation tags to select correct message metadata (types 102, 103, and 574) have validation tags with valid values.
- All mandatory text block sequences and fields are displayed in the message.
- The text block contains no invalid fields.

Limitations and Suggestions

When using the SWIFT MT palette of the plug-in, for message types containing more than one Qualifier Groups, the order of the output MT message being displayed in the Qualifier Groups can be different from that of the input MT message. This is not an error. The Qualifier Groups can be displayed in any order according to the SWIFT specification.

SWIFT MX Palette

The SWIFT MX palette contains two activities, which are used to handle the SWIFT MX message:

- Parse SWIFT MX Activity
- Render SWIFT MX Activity

Parse SWIFT MX Activity

You can use the Parse SWIFT MX activity to validate a SWIFT MX message and parse it to XML format.

General

In the **General** tab, you can select the MX schema to parse the SWIFT MX message.

The following table lists the configurations in the **General** tab of the Parse SWIFT MX activity:

Field	Module Property?	Description
Name	No	Specify the name to be displayed as the label for the activity in the process.
SWIFT Specification	Yes	Click 🔍 to select a Load SWIFT MX Schema shared resource. If no matching Load SWIFT MX Schema shared resource is found, click **Create Shared Resource** to create one. For more details, see Creating a Load SWIFT MX Schema Shared Resource.
Schema	No	The following schemas are available in the **Schema** list: • MX Schema • All Schema
Transport Schema	No	Click 🔍 to select the transport schema that you want to parse. This field is displayed when you select **All Schema** from the **Schema** list.

Field	Module Property?	Description
Element	No	Select a message type from the **Element** list. This field is displayed when you select **All Schema** from the **Schema** list. It provides a list of message element types after you select a transport schema in the **Transport Schema** field. The following items are available in the **Element** list after you select an InterAct type transport schema: • ExchangeRequest • ExchangeResponse • HandleRequest • HandleResponse Only the `DataPDU` item is available in the **Element** list after you select an `SAA type` transport schema.
AppHeader Schema	No	Click 🔍 to select the AppHeader schema that you want to parse. This field is displayed when you select **All Schema** from the **Schema** list.
MX Schema	No	Click 🔍 to select an MX schema which is loaded in the Load SWIFT MX Schema shared resource.
Select All Validations	No	Select this check box to enable all the validations.
Validate Transport Schema	No	Select this check box to enable message transport validation in SWIFT messages against the SWIFT transport schema. This field is displayed when you select **All Schema** from the **Schema** list.
Validate AppHeader Schema	No	Select this check box to enable message header validation in SWIFT messages against the SWIFT AppHeader schema. This field is displayed when you select **All Schema** from the **Schema** list.
Validate MX Schema	No	Select this check box to enable syntax validation in SWIFT messages against the SWIFT MX schema.
Validate MX Rules	No	Select this check box to enable validation for extended rules and business logic in SWIFT messages against the MX rules.

Field	Module Property?	Description
Validate BIC/BEI	No	Select this check box to validate the SWIFT message BIC/BEI codes against a BIC directory. SWIFT BIC and BEI codes are entity identifiers that are displayed in various subfields of the text block. Validating them takes a lot of time and memory, because data must be searched in a very large database. At the initialization, an `FI.dat` BIC directory is provided to the validator, and then full BIC/BEI validation is performed against the provided directory. If no BIC directory is provided, the plug-in uses the default BIC file. Use the latest BIC file provided by SWIFT for up-to-date BIC or BEI validation.
Validate IBAN/BBAN	No	Select this check box to enable IBAN/BBAN validation in SWIFT messages against an IBAN/BBAN directory.
BANK Directory and IBAN Plus	No	Click 🔍 and navigate to the location of the BANK directory and IBAN plus file. Select the file and click **Open** to load it. This field is available when you select the **Validate IBAN/ BBAN** check box.
BIC Code File(FI.dat)	Yes	Click 🔍 and navigate to the location of the BIC or BICPlus file. Select the file and click **Open** to load it. The BIC or BICPlus file is extremely large and, if specified, causes the increase of plug-in startup time. If this file is not specified, the default `FI.dat` file is used.
ISO3166 Country Code File(CT.dat)	Yes	Click 🔍 to navigate to the location of the country code data file. Select the file and click **Open** to load it. The ISO3166 country codes are used if the file is specified here. Otherwise, an internal database of country codes is used.
ISO4217 Currency Code File(CU.dat)	Yes	Click 🔍 to navigate to the location of the currency code data file. Select the file and click **Open** to load it. The ISO4217 currency codes are used if the file is specified here. Otherwise an internal database of currency codes is used.
Validation Filter File(ValidationFilter.xml)	Yes	Click 🔍 to navigate to the location of the `ValidationFilter.xml` file. Select the file and click **Open** to load it. See Using Validation Filters for more information about the validation filters feature.

Description

In the **Description** tab, you can enter a short description for the Parse SWIFT MX activity.

Input

The following table lists the input element in the **Input** tab of the Parse SWIFT MX activity:

Input Item	Data Type	Description
`MXMessage`	String	MX message in String format.

Output

In the **Output** tab, you can find the parse results.

The following table lists the output element in the **Output** tab of the Parse SWIFT MX activity:

Output Item	Data Type	Description
`SWIFTMessage`	XML	MX message in XML format. The contents of the schema are determined by the SWIFT Message schema specified in the **General** tab.

Fault

In the **Fault** tab, you can find the error code and error message of the Parse SWIFT MX activity. See Error Codes for more detailed explanation of errors.

The following table lists the error schema elements in the **Fault** tab of the Parse SWIFT MX activity:

Error Schema Element	Data Type	Description
`ValidationException`	String	The MX messages have validation errors when the validation does not follow the selected SWIFT specification.
`SwiftException`	String	The activity cannot locate the metadata directory or it fails to initialize the metadata.

Render SWIFT MX Activity

You can use the Render SWIFT MX activity to generate specific MX messages.

General

In the **General** tab, you can select the MX schema to render a SWIFT MX message.

The following table lists the configurations in the **General** tab of the Render SWIFT MX activity:

Field	Module Property?	Description
Name	No	Specify the name to be displayed as the label for the activity in the process.
SWIFT Specification	Yes	Click 🔍 to select a Load SWIFT MX Schema shared resource. If no matching Load SWIFT MX Schema shared resource is found, click **Create Shared Resource** to create one. For more details, see Creating a Load SWIFT MX Schema Shared Resource.
Schema	No	The following schemas are available in the **Schema** list: • MX Schema • All Schema
Transport Schema	No	Click 🔍 to select the transport schema that you want to parse. This field is displayed when you select **All Schema** from the **Schema** list.
Element	No	Select a message type from the **Element** list. This field is displayed when you select **All Schema** from the **Schema** list. It provides a list of message element types after you select a transport schema in the **Transport Schema** field. The following items are available in the **Element** list after you select an InterAct type transport schema: • ExchangeRequest • ExchangeResponse • HandleRequest • HandleResponse Only the `DataPDU` item is available in the **Element** list after you select an `SAA type` transport schema.
AppHeader Schema	No	Click 🔍 to select the AppHeader schema that you want to parse. This field is displayed when you select **All Schema** from the **Schema** list.
MX Schema	No	Click 🔍 to select an MX schema which is loaded in the Load SWIFT MX Schema shared resource.
Select All Validations	No	Select this check box to enable all the validations.

Field	Module Property?	Description
Validate Transport Schema	No	Select this check box to enable message transport validation in SWIFT messages against the SWIFT transport schema. This field is displayed when you select **All Schema** from the **Schema** list.
Validate AppHeader Schema	No	Select this check box to enable message header validation in SWIFT messages against the SWIFT AppHeader schema. This field is displayed when you select **All Schema** from the **Schema** list.
Validate MX Schema	No	Select this check box to enable syntax validation in SWIFT messages against the SWIFT MX schema.
Validate MX Rules	No	Select this check box to enable validation for extended rules and business logic in SWIFT messages against the MX rules.
Validate BIC/BEI	No	Select this check box to validate the SWIFT message BIC/BEI codes against a BIC directory. SWIFT BIC and BEI codes are entity identifiers that are displayed in various subfields of the text block. Validating them takes a lot of time and memory, because data must be searched in a very large database. At the initialization, an `FI.dat` BIC directory is provided to the validator, and then full BIC/BEI validation is performed against the provided directory. If no BIC directory is provided, the plug-in uses the default BIC file. Use the latest BIC file provided by SWIFT for up-to-date BIC or BEI validation.
Validate IBAN/BBAN	No	Select this check box to enable IBAN/BBAN validation in SWIFT messages against an IBAN/BBAN directory.
BANK Directory and IBAN Plus	No	Click 🔍 and navigate to the location of the BANK directory and IBAN plus file. Select the file and click **Open** to load it. This field is available when you select the **Validate IBAN/ BBAN** check box.
BIC Code File(FI.dat)	Yes	Click 🔍 and navigate to the location of BANK directory and IBAN plus file. Select the file and click **Open** to load it. The BIC or BICPlus file is extremely large and, if specified, causes the increase of plug-in startup time. If this file is not specified, the default `FI.dat`file is used.

Field	Module Property?	Description
ISO3166 Country Code File(CT.dat)	Yes	Click 🔍 to navigate to the location of the country code data file. Select the file and click **Open** to load it. The ISO3166 country codes are used if the file is specified here. Otherwise, an internal database of country codes is used.
ISO4217 Currency Code File(CU.dat)	Yes	Click 🔍 to navigate to the location of the currency code data file. Select the file and click **Open** to load it. The ISO4217 currency codes are used if the file is specified here. Otherwise an internal database of currency codes is used.
Validation Filter File(ValidationFilter.xml)	Yes	Click 🔍 to navigate to the location of the `ValidationFilter.xml` file. Select the file and click **Open** to load it. See Using Validation Filters for more information about the validation filters feature.

Description

In the **Description** tab, you can enter a short description for the Render SWIFT MX activity.

Input

The following table lists the input element in the **Input** tab of the Render SWIFT MX activity:

Input Item	Data Type	Description
`SWIFTMessage`	XML	MX message in XML format. The contents of the schema are determined by the SWIFT Message schema specified in the **General** tab.

Output

In the **Output** tab, you can find the render results.

The following table lists the output element in the **Output** tab of the Render SWIFT MX activity:

Output Item	Data Type	Description
`MXMessage`	String	MX message in String format.

Fault

In the **Fault** tab, you can find the error code and error message of the Render SWIFT MX activity. See Error Codes for more detailed explanation of the error.

The following table lists the error schema elements in the **Fault** tab of the Render SWIFT MX activity:

Error Schema Element	Data Type	Description
`ValidationException`	String	The MX messages have validation errors when the validation does not follow the selected SWIFT specification.
`SwiftException`	String	The activity cannot locate the metadata directory or it fails to initialize the metadata.

Configuring Advanced Options

This plug-in provides the following advanced configuration options:

- Parsing and Validating SWIFT MT Messages Using SwiftCheck
- Parsing and Validating SWIFT MX Messages Using SwiftMXCheck
- Loading SWIFT MT Message Types into a Repository
- Using Validation Filters
- Configuring Customized MX Java Rules
- Using Dynamic BIC or BEI Update

Parsing and Validating SWIFT MT Messages Using SwiftCheck

You can use the SwiftCheck utility to parse and validate one or more SWIFT MT message files. The utility can be used for low-level testing of SWIFT MT message files.

Procedure

1. Navigate to the `TIBCO_HOME/bw/palettes/swift/version_number/bin` directory.
2. Copy the `SwiftCheck.tra` file, rename the copied file as `mySwiftCheck.tra`, and open it in the text editor.
3. Change the **`application.args`** property to include the metadata and validation options to use. For example:

   ```
   application.args -data TIBCO_HOME/bw/palettes/swift/version_number/bin/data -set
   specification_year -v TIBCO_HOME/bw/palettes/swift/samples/SampleMT.txt
   ```

   For more information about the options, see SwiftCheck Options.

4. Save the `mySwiftCheck.tra` file and exit the text editor.
5. Open a command line and change to the `TIBCO_HOME/bw/palettes/swift/version_number/bin` directory:

   ```
   cd TIBCO_HOME/bw/palettes/swift/version_number/bin
   ```

6. Type the following command to start the utility:

   ```
   SwiftCheck --run --propFile mySwiftCheck.tra
   ```

SwiftCheck Options

You can use the following options to parse and validate the SWIFT MT message files.

Metadata Options

The following table shows the descriptions of the metadata options:

Option	Description
`-data or --dataDir <file>`	metadata directory
`-set or --dataSet <file>`	metadata set
`-fi or --fiFile <file>`	bankdata file (BIC codes)

Option	Description
`-ff or --filterFile <file>`	validation warning filter file
`-cu or --cuFile <file>`	currency code file
`-ct or --ctFile <file>`	country code file

Validation options

The following table shows the description of the validation options:

Option	Description
-v or --valAll	validate using all rules
-vc or --valFormat	turn on format validation
-vt or --valType	turn on data type validation
-vw or --valWord	turn on code word validation
-vb or --valBic	turn on BIC code validation
-vf or --valField	turn on field level semantics validation
-vm or --valMsg	turn on message level semantics validation
-vn or --valNarr	turn on narrative validation
-va or --valAdvanced	turn on advanced header validation
-vs or --valStructural	turn on structural validation

Parsing and Validating SWIFT MX Messages Using SwiftMXCheck

You can use the SwiftMXCheck utility to validate one or more SWIFT MX message files. This utility can be used for low-level testing of SWIFT MX message files.

Procedure

1. Navigate to the `TIBCO_HOME/bw/palettes/swift/version_number/bin` directory.
2. Copy the `SwiftMXCheck.tra` file, rename the copied file as `mySwiftMXCheck.tra`, and open it in the text editor.
3. Change the **`application.args`** property to include the metadata and validation options to use. For example:

   ```
   application.args

   -mxdata TIBCO_HOME/bw/palettes/swift/version_number/bin/mxdata -set year

   -txsd TIBCO_HOME/bw/palettes/swift/version_number/bin/xsd/year/SwInt.xsd

   -axsd TIBCO_HOME/bw/palettes/swift/version_number/bin/xsd/year/$ahV10.xsd

   -mxsd TIBCO_HOME/bw/palettes/swift/version_number/bin/xsd/year/ -ib IBAN_dir
   ```

```
-v TIBCO_HOME/bw/palettes/swift/version_number/samples/SampleMX.xml
```

For more information about the options, see SwiftMXCheck Options.

4. Save the `mySwiftMXCheck.tra` file and exit the text editor.
5. Open a command line and change to the `TIBCO_HOME/bw/palettes/swift/version_number/bin` directory:

   ```
   cd TIBCO_HOME/bw/palettes/swift/version_number/bin
   ```

6. Type the following command to start the utility:

   ```
   SwiftMXCheck --run --propFile mySwiftMXCheck.tra
   ```

SwiftMXCheck Options

You can use the following options to validate the SWIFT MX message files.

Metadata Options

The following table shows the description of the metadata options:

Option	Description
`-mxdata or --mxdataDir <file>`	metamxdata directory
`-set or --mxdataSet <file>`	mxmetadata set
`-schema or --useSchema <Int>`	Schema
`-txsd or --txsdFile <file>`	Transport XSD File
`-axsd or --axsdFile <file>`	AppHdr XSD File
`-mxsd or --mxsdFile <file or dir>`	MX XSD File or Dir
`-ib or --ibDir <dir>`	IBAN/BBAN Dir
`-fi or --fiFile <file>`	bankdata file (BIC codes)
`-cu or --cuFile <file>`	currency code file
`-ct or --ctFile <file>`	country code file
`-ff or --filterFile <file>`	validation warning filter file

Validation Options

The following table shows the description of the validation options:

Option	Description
-v or --valAll	validate using all rules
-vt or --valTransportSchema	turn on Transport schema validation
-va or --valAppHdrSchema	turn on AppHdr schema validation

Option	Description
-vm or --valMXSchma	turn on MX schema validation
-vr or --valMXrules	turn on MX rules validation
-vb or --valBic	turn on BIC code validation
-vi or --valIBAN	turn on IBAN/BBAN code validation

Loading SWIFT MT Message Types into a Repository

You can use the SwiftSchemaLoader utility to load SWIFT MT message types into a repository.

Procedure

1. Navigate to the `TIBCO_HOME/bw/palettes/swift/version_number/bin` directory.
2. Copy the `SwiftSchemaLoader.tra` file, rename the copied file as `mySwiftSchemaLoader.tra`, and open it in the text editor.
3. Change the **`application.args`** property with the repository location, the plug-in home directory, SWIFT specification, load or unload (-l,-u) and message types. For example:

   ```
   application.args

   -system:propFile TIBCO_HOME/bw/palettes/swift/version_number/bin/
   mySwiftSchemaLoader.tra

   -p C:/workspace/test_project

   -h TIBCO_HOME/bw/palettes/swift/version_number -s specification_year -l -m 517
   ```

   Ensure you have created a project before loading the SWIFT MT message types.
4. Save the `mySwiftSchemaLoader.tra` file and exit the text editor.
5. Open a command line and change to the `TIBCO_HOME/bw/palettes/swift/version_number/bin` directory:

   ```
   cd TIBCO_HOME/bw/palettes/swift/version_number/bin
   ```
6. Type the following command to start the utility:

   ```
   SwiftSchemaLoader --run --propFile mySwiftSchemaLoader.tra
   ```

SwiftSchemaLoader Options

You can use the following options to load the SWIFT message types.

```
-p the source project. The project can be local or remote

-h: Swift home directory

-s: Specification.<For example:2005>

-l/-u Load/Remove messages

-m: Message types separated by comma(101,517,ect. or all for loading/unloading all
messages)
```

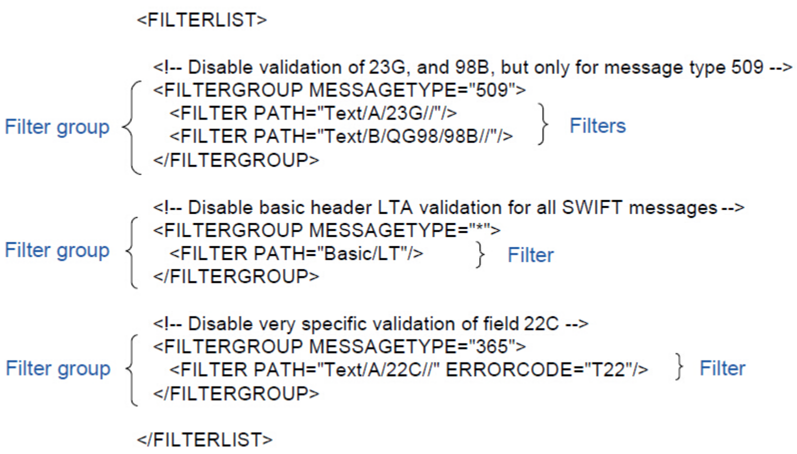
Using Validation Filters

When processing a message, you can customize a validation filter file to ignore some specified validation errors.

Filter Groups

The `ValidationFilter.xml` file consists of a list of filter groups. Each filter group has zero or more filters.

The following figure shows an example list of MT filter groups in the `ValidationFilter.xml` file:



```
<FILTERLIST>

  <!-- Disable validation of 23G, and 98B, but only for message type 509 -->
  <FILTERGROUP MESSAGETYPE="509">
    <FILTER PATH="Text/A/23G//"/>
    <FILTER PATH="Text/B/QG98/98B//"/>
  </FILTERGROUP>

  <!-- Disable basic header LTA validation for all SWIFT messages -->
  <FILTERGROUP MESSAGETYPE="*">
    <FILTER PATH="Basic/LT"/>
  </FILTERGROUP>

  <!-- Disable very specific validation of field 22C -->
  <FILTERGROUP MESSAGETYPE="365">
    <FILTER PATH="Text/A/22C//" ERRORCODE="T22"/>
  </FILTERGROUP>

</FILTERLIST>
```

For more details, see the `ValidationFilter.xml` file in the *`TIBCO_HOME`*`/bw/palettes/swift/`*`version_number`*`/samples/ValidationFilter_for_MT` or *`TIBCO_HOME`*`/bw/palettes/swift/`*`version_number`*`/samples/ValidationFilter_for_MX` directory.

Each filter group has a specification that defines the SWIFT message type or types to which the group's filters apply. A wildcard "*" character is permitted as the entire message type specification, or can be used after one or more message type characters. The specification "*", for example, applies its filter group to all SWIFT message types; the specification "5*" applies its filter group to all SWIFT message types whose first digit is 5; the specification "sese.023.001*" applies its filter group to all SWIFT message types whose first prefix is sese.023.001, and so on.

Filters

Each filter group has zero or more filters. With these filters, you can have access to even finer-grained control of validation.

Filtering does not apply to parse errors, only to validation errors. If a SWIFT message is invalid at a fundamental structural level and cannot be parsed, such parse errors cannot be filtered out.

Filters for MT messages are different from those for MX messages.

MT Filters

Each MT filter contains a "disable path specification" and an optional error code.

A path specification is an XPath-like string that identifies one or more fields in a SWIFT MT message. See Path Specification for details on how to construct path specifications. All fields that match the MT filter's path specification have their validation warnings ignored.

If an MT filter also contains a SWIFT error code, then in addition to a path match with a field, only those validation errors with that particular error code are ignored. With this feature, you can have access to even finer-grained control of validation.

Consider the following filter for MT messages:

```
<FILTER PATH="Text/A/22C//"/>
```

The preceding filter ignores validation errors for all occurrences of field 22C in sequence A, for all messages in the filter's filter group. Another example:

```
<FILTER PATH="Text/A/22C//" ERRORCODE="T22"/>
```

This filter is more fine-grained than the previous filter. It only ignores validation errors on field 22C in sequence A with SWIFT error code T22. All other validation errors for this field pass through the filter.

Not all MT filters have to apply to the text block of a SWIFT MT message. The following filter disables validation of the basic header's Logical Terminal Address:

```
<FILTER PATH="Basic/LT"/>
```

The following XML contains examples of how to disable various kinds of validation:

```
<FILTERLIST>
<!--Disable validation of 23G, and 98B, but only for message type 509 -->
<FILTERGROUP MESSAGETYPE="509">
<FILTER PATH="Text/A/23G//" />
<FILTER PATH="Text/B/QG98/98B//" />
</FILTERGROUP>
<!-- Disable basic header LTA validation for all SWIFT messages-->
<FILTERGROUP MESSAGETYPE="*">
<FILTER PATH="Basic/LT" />
</FILTERGROUP>
<!-- Very specific validation of field 22C -->
<FILTERGROUP MESSAGETYPE="365">
<FILTER PATH="Text/A/22C//" ERRORCODE="T22" />
</FILTERGROUP>
</FILTERLIST>
```

For more details, see the `ValidationFilter.xml` file in the *TIBCO_HOME*/bw/palettes/swift/*version_number*/samples/ValidationFilter_for_MT directory.

Path Specification

An MT filter uses a subset of the XPath grammar with a few additional abbreviations to identify one or more elements of a message.

The XPath like syntax for the identifying path is composed of tokens separated by the slash (/) character. Each token is composed of an element name or the wildcard character (*). Optionally a predicate can be added inside square brackets immediately following the element name. The predicates supported are a simple one based index to identify the position of an element and an expression to identify an attribute of the element. You can only use one predicate in square brackets for each element name. In addition, only tag fields support the attribute value expression. You can use an additional abbreviation with the attribute matching expression. Multiple values can be entered separately by a vertical bar to denote that either of the supplied values must match.

The slash separator character is used to mark the separation of element names. They are also used to identify characteristics of the start and end of the path. A slash at the start of a path identifies the path as an absolute path. Since all paths are evaluated from the message, the path can be a relative path that does not start with a slash or an absolute path that does start with a slash. As an addition to XPath, a slash at the end of the path indicates that the path must match the last name in the path as well as any immediate children of that element. Two slashes indicate all descendents of the element must match.

The following table shows the description of the path specification abbreviations:

Field	Description
Text	The message Text block as sequences
Basic	The message Basic Header: AppId, Service, LT, Session, Sequence

Field	Description
App	The message Application Header: • io: IO flag, either "I" or "O". • I: Input Header: Type, Receiver, Priority, Delivery, Obsolescence. • O: Output Header: Type, InputTime, MIR, OutputDate, OutputTime, Priority.
User	The message User Header
Tags	The message Text block as a flat list of tag fields
Trailer	The message Trailer
S	The message SWIFT Alliance trailer if it exists

Valid Paths Example

The following table lists examples of valid paths:

Example	Description
`/Basic/AppId`	The AppId in the Basic Header
`/Basic/*`	Any subfield of the Basic Header
`Basic/*[2]`	The second subfield in the Basic Header
`Basic/2`	The second subfield in the Basic Header
`/Text/A/A1[2]/A1a/16R/1`	The first subfield of a 16R
`/Text/A/*[2]/16R/Qualifier`	Any second sequence in A
`Text/A/*/57D/2-3`	Subfield 2 to 3 of tag field 57D in any sequence in A
`/Tags/16R/Qualifier`	Used to identify the subfield by name
`Tags/*/1`	The first subfield of the any tag field
`/App/O/MIR`	MIR in the Output Header of the Application Header
`/App/io`	The io value for the Application Header
`/User/119`	Subfield 119 of the User Header
`Trailer/MAC`	Subfield MAC of the Trailer
`/Text/A/`	A and any immediate sequences, qualifier groups, or fields

Example	Description
`Text/A//`	A and any descendent sequences, qualifier groups, or fields
`/Text/A/A1[2]/QG22/22F [@Qualifier=MICO]`	A 22F tag field where the Qualifier subfield is 'MICO'
`Text/A/A1[2]/QG22/22F [@Qualifier=MICO|FORM]`	A 22F tag field where the Qualifier subfield is 'MICO' or 'FORM'
`//`	The message and any element in the message

Invalid Paths

The following table lists the invalid paths:

Field	Description
`Text/A/A1[2]/A1a/16RS/ Qualifier`	Multiple option characters for tags with different field names are not supported.
`Invalid/12A/2`	Invalid part of the message.

MX Filters

Each MX filter contains a SWIFT error code. All MX messages that match the error code of the MX filter have their validation warnings ignored.

The following code is an example of MX validation filter:

```
<FILTERLIST>
<!-- Disables validation all sese.023.001* message's error code D00008 -->
<FILTERGROUP MESSAGETYPE="sese.023.001*">
<FILTER ERRORCODE="D00008"/>
</FILTERGROUP>
</FILTERLIST>
```

This filter ignores validation errors in MX messages with the prefix sese.023.001* and SWIFT error code D00008. All other validation errors pass through the filter.

For more details, see the `ValidationFilter.xml` file in the *`TIBCO_HOME`*`/bw/palettes/swift/`*`version_number`*`/samples/ValidationFilter_for_MX` directory.

Configuring Customized MX Java Rules

You can configure customized MX Java rules for the plug-in to support additional MX message type schemas.

Procedure

1. Open a code writing software to write the Java code. See MX Java Rule Code Example for more information.

When writing an MX Java rule code:

- You have to extend the MXValidationRule Java class and override eval() method.
- The Java class name must be the same as the rule name in the SWIFT Standard MX file.
- The Java rule code depends on the `dom4j-1.6.1.jar` file and `com.tibco.bw.palette.swift.common_6.1.0.0xx.jar`, which are located in the `TIBCO_HOME/bw/palettes/swift/version_number/tools/lib` and `TIBCO_HOME/bw/palettes/swift/version_number/runtime/plugins` directory.

2. After writing the Java code, export the Java code to a JAR file and save it to the `TIBCO_HOME/bw/palettes/swift/version_number/lib` directory.
3. Add a customized MX message type to the `KnownMessages.xml` file, which is located in the `TIBCO_HOME/bw/palettes/swift/version_number/bin/mxdata/year` directory.
4. Configure a customized MX message rule in a specific XML file in the `TIBCO_HOME/bw/palettes/swift/version_number/bin/mxdata/year` directory based on the message type.

The prefix of the MX message type name in an MX message rule must be the same as the suffix of a specific XML file in the `TIBCO_HOME/bw/palettes/swift/version_number/bin/mxdata/year` directory. For example:

- The `semt.021.001.05` MX message rule must be configured in the `MXsemt.xml` file.
- The `pacs.003.001.05` MX message rule must be configured in the `MXpacs.xml` file.
- The `acmt.001.001.05` MX message rule must be configured in the `MXacmt.xml` file.

For detailed information about an example of the MX message rule, see MX Message Rule Example.

5. Restart TIBCO Business Studio.

Result

The customized MX Java rule becomes available, and the plug-in supports the corresponding MX message.

MX Java Rule Code Example

The following code is an example of MX Java rule code.

```
package com.customer.swift.spec2015;
import org.dom4j.Element;
import com.tibco.swift2.msg.MXFinder;
import com.tibco.swift2.msg.SwiftMXMessage;
import com.tibco.swift2.util.SwiftException;
import com.tibco.swift2.validate.ArgResolver;
import com.tibco.swift2.validate.MXValidationRule;
import com.tibco.swift2.validate.MXValidator;
public class OtherIdentificationPresenceRule extends
MXValidationRule {
public OtherIdentificationPresenceRule(String name,String xpath, String
overrideErrorSeverity, String overrideErrorCode,String overrideErrorText,
String[] args, ArgResolver resolver) throws SwiftException {super(name, xpath,
overrideErrorSeverity, overrideErrorCode,overrideErrorText,args, resolver, 0, 0,
new Class[0]);}
@Override
public boolean eval(SwiftMXMessage msg, MXValidator validator, int validationLevel)
{
MXFinder finderFinInstrmId=new
MXFinder(getParam_FinInstrmId());
MXFinder finderISIN= new MXFinder(getParam_ISIN());
MXFinder finderOthrId= new MXFinder(getParam_OthrId());
MXFinder finderDesc= new MXFinder(getParam_Desc());
boolean isValid = true;
Element elementISIN=finderISIN.getElement(msg);
Element elementDesc=finderDesc.getElement(msg);
Element elementOthrId=finderOthrId.getElement(msg);
Element elementFinInstrmId=finderFinInstrmId.getElement(msg);
if(elementOthrId!=null)
return true;
if(elementFinInstrmId!=null&&elementISIN==null&&elementDesc==null) {
msg.addWarning(elementFinInstrmId, getName(),
getOverrideErrorCode(),getOverrideErrorText());
isValid = false;
}
return isValid;
}
public String getParam_FinInstrmId(){
return getParam().get("FinInstrmId");
}
public String getParam_OthrId(){
return getParam().get("OthrId");
}
public String getParam_ISIN(){
return getParam().get("ISIN");
}
public String getParam_Desc(){
return getParam().get("Desc");
}
}
```

MX Message Rule Example

The following code is an MX message rule example.

```
<MESSAGE TYPE="semt.021.001.05"
NAME="SecuritiesStatementQueryV05">
<RULE NAME="C1" EXPRESSION="AnyBICRule()" XPATH="//AnyBIC" ERRORTEXT="Invalid BIC"
ERRORCODE="D00008"/>
<RULE NAME="C7" EXPRESSION="spec2015.DescriptionPresenceRule()" ERRORTEXT="At least
one identification must be present" ERRORCODE="X00192">
<PARAM NAME="FinInstrmId" XPATH="//AddtlQryParams//FinInstrmId"/>
<PARAM NAME="ISIN" XPATH="//FinInstrmId/ISIN"/>
<PARAM NAME="OthrId" XPATH="//FinInstrmId/OthrId"/>
<PARAM NAME="Desc" XPATH="//FinInstrmId/Desc"/>
</RULE>
<RULE NAME="C10" EXPRESSION="spec2015.ISINPresenceRule()" ERRORTEXT="At least one
identification must be present" ERRORCODE="X00194">
<PARAM NAME="FinInstrmId" XPATH="//AddtlQryParams//FinInstrmId"/>
<PARAM NAME="ISIN" XPATH="//FinInstrmId/ISIN"/>
<PARAM NAME="OthrId" XPATH="//FinInstrmId/OthrId"/>
<PARAM NAME="Desc" XPATH="//FinInstrmId/Desc"/>
</RULE>
<RULE NAME="C11"
EXPRESSION="com.customer.swift.spec2015.OtherIdentificationPresenceRule()"
ERRORTEXT="At least one identification must be present" ERRORCODE="X00193">
<PARAM NAME="FinInstrmId" XPATH="//AddtlQryParams//FinInstrmId"/>
<PARAM NAME="ISIN" XPATH="//FinInstrmId/ISIN"/>
<PARAM NAME="OthrId" XPATH="//FinInstrmId/OthrId"/>
<PARAM NAME="Desc" XPATH="//FinInstrmId/Desc"/>
</RULE>
</MESSAGE>
```

Using Dynamic BIC or BEI Update

The plug-in supports the BIC or BEI validations. If changes are made to the BIC file, you can use the TIBCO Enterprise Message Service™ message or TIBCO Rendezvous® message to update BIC or BEI validations dynamically.

The following tasks demonstrate how to use the dynamic BIC or BEI update feature:

1. Activating Dynamic BIC or BEI Update
2. Triggering Dynamic BIC or BEI Update

Activating Dynamic BIC or BEI Update

You have to activate the dynamic BIC or BEI update feature before using it.

Procedure

1. Navigate to the `TIBCO_HOME/bw/version_number/domains/domain_name/appnodes/appspace_name/appnode_name` directory.

   Ensure that you have created a domain before activating the dynamic BIC or BEI update feature.

2. Open the `config.ini` file, and add the following parameter:

   BIC_UPDATE_CONFIG_FILE_PATH=`TIBCO_HOME/bw/palettes/swift/version_number/samples/BICUpdate.properties`.

3. Save the file.

Triggering Dynamic BIC or BEI Update

The BIC information changes if you edit or overwrite the `FI.dat` file. You can generate a trigger manually or automatically to indicate to the plug-in the changes made to the file.

Manually

Using this method, you have to send a message to the plug-in using TIBCO Enterprise Message Service or TIBCO Rendezvous.

Automatically

In TIBCO BusinessWorks Studio, configure a process with the following configurations:

- Poll the BIC file by using the File Poller activity to identify the changes in its content.
- Configure the TIBCO Enterprise Message Service or TIBCO Rendezvous publisher activity for the required subject, topic, or queue.

To receive the trigger, the plug-in updates its temporary memory (in-memory cache) with new BIC information. The following table lists the trigger types and parameters:

Trigger Type	Parameter	Notes
TIBCO Enterprise Message Service (EMS)	Server parameters: **EMS_Server_URL** **EMS_Username** **EMS_Password** Destination parameters: **EMS_DestinationName** **EMS_DestinationType(Topic or Queue)**	If you update the BIC file, you have to send a TIBCO Enterprise Message Service message on the configured topic or queue that has the BIC file name as the message body.

Trigger Type	Parameter	Notes

TIBCO Rendezvous (RV)	TIBCO Rendezvous parameters:
	`RV_Service`
	`RV_Network`
	`RV_Daemon`
	`RV_Subject`
		If you update the BIC file, you have to send a TIBCO Rendezvous message on the configured subject that has the BIC file name as the message body.

Processing Acknowledgment Messages

The acknowledgment is sent by the MT service to Logical Terminal to confirm the receipt of a message and its safe storage by the service.

If an ACK is returned, the message is accepted by the MT service for delivery to its destination. If a NAK is returned, the message is safely stored, but failed to delivery. The structure of an ACK or a NAK message is the same, except that the value of tag 451 in the text block of an Acknowledgment message. 0 indicates the ACK, and 1 indicates the NAK .

MT acknowledgment messages are parsed using the FIN_Acknowledgment schema. This schema loads automatically after you create an Load SWIFT MT schema shared resource. The plug-in supports the parsing and rendering of the ACK and NAK message.

Procedure

1. Open the process that you want to process the acknowledge messages.

   > Ensure that you have created an Load SWIFT MT schema shared resource in this process.

2. Click the **General** tab of the Parse SWIFT MT activity or Render SWIFT MT activity.
3. In the **SWIFT Message Schema** field, click 🔍 and select the `Message_FIN_Acknowledgement` message schema.
4. Save the process.

Reconciling Acknowledgment Messages

The acknowledgment message from the SWIFT network contains the field 108. The field carries Message User Reference (MUR) information of the original message. This information can be presented in one of the following ways:

- MUR information is present in the user header of the original message.
- If no MUR is present in the original message, contents of Field 20 of the original message or (for Category 5 messages only) the contents of Field 20C, with the code word SEME, but only when all the letters of the alphabets are uppercase.
- Contents of Field 20C.

For reconciling, you must have the reference information of the MT message sent and value of the tag 108 of the acknowledgment message received. Compare both information. If they match, the reconciliation is successful.

Managing Logs

When an error occurs, you can check logs to trace and troubleshoot the plug-in exceptions.

By default, error logs are displayed in the Console view when you run a process in the debug mode. You can change the log level of the plug-in to trace different messages and export logs to a file. Different log levels correspond to different messages, as described in Log Levels.

Setting Up Log Levels

You can configure a different log level for the plug-in and plug-in activities to trace different messages.

By default, the plug-in uses the log level configured for TIBCO ActiveMatrix BusinessWorks. The default log level of TIBCO ActiveMatrix BusinessWorks is `Error`.

Procedure

1. Navigate to the `TIBCO_HOME/bw/version_number/config/design/logback` directory and open the `logback.xml` file.
2. Add the following node in the **BusinessWorks Palette and Activity loggers** area to specify a log level for the plug-in.

   For the activities in the SWIFT MT palette, add the following node:

   ```
   <logger name="com.tibco.bw.palette.swift.runtime">
      <level value="DEBUG"/>
   </logger>
   ```

   For the activities in the SWIFT MX palette, add the following node:

   ```
   <logger name="com.tibco.bw.palette.swiftmx.runtime">
      <level value="DEBUG"/>
   </logger>
   ```

   The value of the **`level`** element can be `Error`, `Info`, or `Debug`.

   If you set the log level to `Debug`, the input and output for the plug-in activities are also displayed in the Console view. See Log Levels for more details regarding each log level.
3. Optional: Add the following node in the **BusinessWorks Palette and Activity loggers** area to specify a log level for an activity:

   For the activities in the SWIFT MT palette, add the following node:

   ```
   <logger name="com.tibco.bw.palette.swift.runtime.ActivityNameActivity">
      <level value="DEBUG"/>
   </logger>
   ```

   For each activity, the *ActivityName* is:

   - Parse SWIFT MT: Parser
   - Render SWIFT MT: Renderer
   - Route SWIFT MT: Router
   - Generate SWIFT BICPlusIBAN: BICPlusIBANGenerator
   - Validate SWIFT BICPlusIBAN: BICPlusIBANValidator

   For the activities in the SWIFT MX palette, add the following node:

   ```
   <logger name="com.tibco.bw.palette.swiftmx.runtime.ActivityNameActivity">
      <level value="DEBUG"/>
   </logger>
   ```

For each activity, the *ActivityName* is:

- Parse SWIFT MX: Mxparser
- Render SWIFT MX: Mxrenderer

For example, add the following node to set the log level of the Parse SWIFT MT activity to `Debug`:

```
<logger name="com.tibco.bw.palette.swift.runtime.ParserActivity">
   <level value="DEBUG"/>
</logger>
```

The activities that are not configured with specific log levels use the log level configured for the plug-in.

4. Save the file.

Log Levels

Different log levels include different information.

The plug-in supports the following log levels:

Log Level	Description
Trace	Includes all information regarding the running process.
Debug	Indicates a developer-defined tracing message.
Info	Indicates normal plug-in operations. No action is required. A tracing message tagged with Info indicates that a significant processing step is reached, and logged for tracking or auditing purposes. Only info messages preceding a tracking identifier are considered as significant steps.
Warn	Indicates that an abnormal condition occurred. Processing continues, but special attention from an administrator is recommended.
Error	Indicates that an unrecoverable error occurred. Depending on the severity of the error, the plug-in might continue with the next operation or might stop.

Exporting Logs to a File

You can update the `logback.xml` file to export plug-in logs to a file.

Procedure

1. Navigate to the *TIBCO_HOME*/bw/*version_number*/config/design/logback directory and open the `logback.xml` file.

   After deploying an application in TIBCO Enterprise Administrator, navigate to the *TIBCO_HOME*/bw/*version_number*/domains/*domain_name*/appnodes/*space_name*/*node_name* directory to find the `logback.xml` file.

2. Add the following node to specify the file where the log is exported:

```
<appender name="FILE" class="ch.qos.logback.core.FileAppender">
   <file>c:/bw6-swift.log</file>
      <encoder>
        <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36}-%msg%n</pattern>
      </encoder>
</appender>
```

   The value of the **`file`** element is the absolute path of the file that stores the exported log.

3. Add the following node to the `root` node at the bottom of the `logback.xml` file:

   ```
   <logger name="com.tibco.bw.palette.swift">
   <appender-ref ref="STDOUT" />
   <appender-ref ref="FILE" />
   <level value="DEBUG"/>
   </logger>
   ```

4. Save the file.

Error Codes

The following table lists error codes, detailed explanation of each error, and where applicable, ways to solve different errors.

Error Code and Error Message	Role	Category	Description	Solution

ERROR_FORMAT2.errorCode=500004
`ERROR_FORMAT2={0}{1}{2}`
	errorRole	BW-Plug-in	When an MT or MX message does not follow the SWIFT specification, or an activity cannot locate the metadata directory or fails to initialize the metadata.	Check errors listed in the error message, and take an appropriate action.

ERROR_FORMAT3.errorCode=500005
`ERROR_FORMAT3={0}{1}`
	errorRole	BW-Plug-in	After the validation of an MT or MX message, an error occurs when you transfer the message.	Check errors listed in the error message, and take an appropriate action.