Visual Composer for SAP NetWeaver Composition Environment - Connectors

Applies to:

Visual Composer for SAP enhancement package 1 for SAP NetWeaver Composition Environment 7.1 For more information, visit the Portal and Collaboration homepage.

Summary

Connectors are points in the data flow that represent connections that channel data from/to points outside the component. This guide explains the different types of connectors Visual Composer has and how to use them.

Author: Yaniv Cohen

Company: SAP

Created on: July 2008

Author Bio

Yaniv is a Quality Engineer working for the Visual Composer group.

Table of Contents

Overview

Connectors are points in the data flow that represent connections that channel data from/to points outside the component.

You can use connectors to:

- Bring parameters in from another component
- Send parameters at the end of the data flow to another component
- Define the component initialization or end value
- Enhance the data flow within the component

You add connectors to your model by dragging them from the Compose task panel to the Design board. You define their logic, flow, and connections to other elements on the Design board.

The following table lists the connectors that are available when building your composite view or service component:

Name	Icon	Available in	Description
Start Point	(Start Point)	Composite view	Defines the start point of the component execution
End Point	(End Point)	Composite view	Defines the end point of the component execution
Navigate	(Navigate)	Composite view	Navigates to another component inside the window
Signal In	(Signal In)	Composite view	Receives a signal from another component
Signal Out	(Signal Out)	Composite view	Sends a signal to another component
Data In	(Data In)	Service component	Receives data from another component
Data Out	(Data Out)	Service component	Sends data to another component
User Data	(User Data)	Composite view Service component	A connector that contains the personal data of the user
Data Store	(Data Store)	Composite view	A signal element that temporarily stores data for use throughout the model
Data Share	(Data Share)	Composite view	Stores complex data structure that can be used by many components in the model
Static Data	(Static Data)	Service component	Stores static data for later use

This document first describes the connectors you can use when working on a composite view model, and then describes the connectors you can use when composing a service component? model.

Composite View Connectors

1. Start Point

Defines a value used to initialize the component execution. For example, in a component used to display a number of customer records, the initialization value defined in the Start Point could be 10.

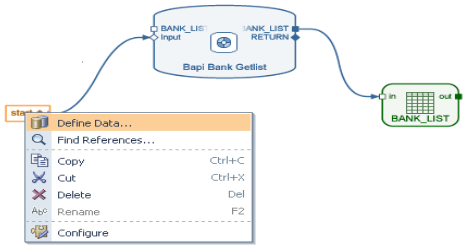
Each model has only one logical starting point. To avoid many connecting lines going out of the starting point to many objects in the model, you can place several Start Points inside your model; all of them point automatically to one logical Start Point.

To demonstrate this capability, we create a simple application with a Start Point, which enables the user to define the Initial Data of the fields BANK_CTRY & MAX_ROWS. The Start Point submits the input values to the data service and triggers the application.

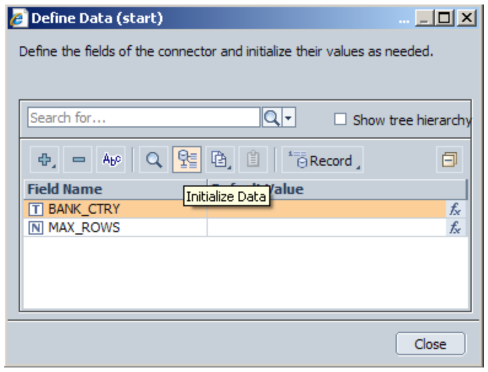
Create and initialize a Start Point:

In this example, the model's Start Point is initialized with data and connected to the Data Service.
To do so, perform the follow the instructions:

a. From the Start Point context menu, select *Define Data*:



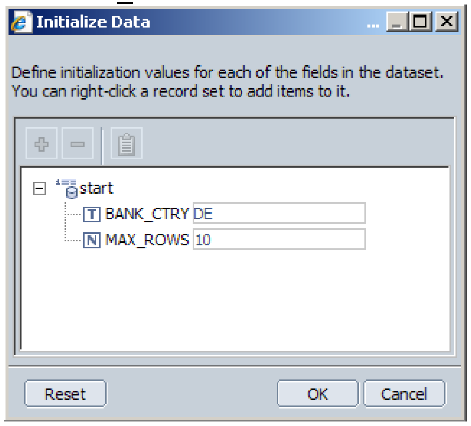
b. Choose the *Initialize Data* button:


Define Data (start)

Define the fields of the connector and initialize their values as needed.

Field Name	Default Value
BANK_CTRY	
MAX_ROWS	

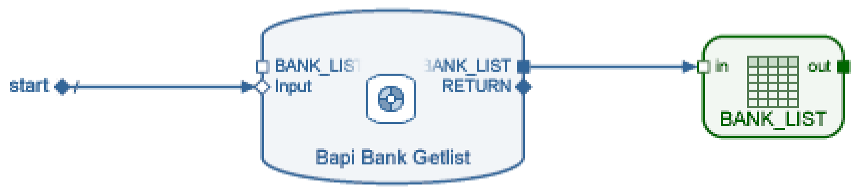
c. Define the value 'DE' in the field 'BANK_CTRY' and the value '10' in the field 'MAX_ROW'


Initialize Data

Define initialization values for each of the fields in the dataset.
You can right-click a record set to add items to it.

start
BANK_CTRY DE
MAX_ROWS 10

d. The model should look similar to the following:



e. When you run the application, the Start Point initializes the data and the submit action automatically sends? the input values to the Data Service and triggers the application.

2. End Point

Defines a final value at the end of the component execution and terminates the execution of the model. For example, in a component that provides a value to another component, the value defined in the End Point is provided to the other Visual Composer model or application that consumes the field.

You create one End Point per component. To do so, you can create a single End Point instance in the model, or, for convenience purposes, you can create multiple End Point elements, all pointing to the same End Point connector.

3. Navigate

Defines the navigation to a referenced component (model), which is open in the main window (replacing the content from the initial component). A component accessed using the Navigate connector cannot output parameters to other components.
When you add a Navigate connector, the *Create New Model* dialog box opens, so you can create the referenced component. When you use a Navigate connector, you must create a new referenced component (you cannot reference an existing component). You can double-click the Navigate connector on the Design board to open the referenced component in a new model tab.

4. Signal In

Receives data sent from another component to a nested component.
Use the Signal In connector to get data from a parent model flowing into a nested model.

Note: After you add a new Signal In connector to a nested component, right-click the nested component icon in the parent component and choose Redefine Ports from the context menu. In the displayed dialog box, select the newly added port and click OK to display the in port on the nested component icon.

5. Signal Out

Sends data from a nested component to another (embedding) component.

Note: After you add a Signal Out connector to a nested component, right-click the nested component icon in the parent component and choose Redefine Ports from the context menu. In the displayed dialog box, select the newly added ports and click OK to display the out port on the nested component icon.

6. User Data

Contains the personal data of the user, retrieved from the runtime environment.
The User Data fields can be referenced in any dynamic expression, thereby personalizing the component by creating dynamic attributes that depend on the current user data and preferences.
For example, you might use the User Data connector as input to a Form View.

7. Data Store

Data Store is a central data container in which you store data for later use. The data is stored in memory, and is available at runtime as long as the modeled application is open. When the modeled application is closed, the data store is emptied.

Each model uses a single Data Store. To make it easier to connect elements to the Data Store, you can add multiple Data Store elements to the Design board. However, these multiple Data Store elements all represent the same Data Store (data stored in one element is visible in the other elements).

The connecting lines to the input port of the Data Store are of data mapping type. The values in the Data Store are accessed using expressions.

8. Data Share

The Data Share connector is usually used with complex data, for two main purposes:

- To maintain a single copy of a dataset for use by multiple elements within a complex model. This is achieved by setting the Scope property of the data share to Private (in the Configure task panel). Unlike the Data Store connector, you can have multiple instances of the data share within a single model. In addition, the records stored in the data share are deleted upon execution of every mapping into that data share.
- Define bi-directional data sharing of complex datasets between Visual Composer components or between a Visual Composer component and another Composition-tool component (such as a Guided Procedures process). This is achieved by setting the Scope property of the data share to Public.

When the source data of the data share is modified, the changes are reflected automatically in the connected components. You can add data to a Data Share by:

- Connecting it to a data service or other model element
- Using the Define Data dialog box for the Data Share

Note: The dataset of the Data Share connector must have the same metadata and data type as the dataset of the element to which it is connected. Visual Composer does not check that the datasets match, so you must manually check the data types before you create the connection.

Service Component Connectors

1. Data In

Receives data from another component (much like Signal In).

Note: After you add a Data In connector to a service component that is referenced from another component, right-click the service component icon in the parent component and choose Redefine Ports from the context menu. In the displayed dialog box, select the newly added ports and click OK to display the in port on the service component icon.

2. Data out

Sends data from a service component to another component (much like Signal Out).

Note: After you add a Data Out connector to a service component that is referenced from another component, right-click the service component icon in the parent component and choose Redefine Ports from the context menu. In the displayed dialog box, select the newly added ports and click OK to display the out port on the service component icon.

3. Static Data

Stores static data for use by a service component. You can create multiple Static Data connectors per service component. You manually define data for a Static Data connector and use it to output data from the service. Unlike a Data Store connector, you cannot add data to the Static Data connector at runtime (the data is redefined and static). When you use the Service Component Wizard to add data to a service component, a Static Data connector is created for each data selection that you add.

Related Content

A Step by Step Guide for Creating a Basic Visual Composer Application

Visual Composer for SAP NetWeaver CE 7.1 SDN page

Visual Composer for NetWeaver CE: Getting Started with a Typical Workflow

For more information, visit the Portal and Collaboration homepage.

Copyright