HP Service Activator

XMaps

Edition: V70-1A

for Microsoft Windows® Server 2012 R2, HP-UX 11i v3,
Red Hat Enterprise Linux 6.6

Manufacturing Part Number: None

January 18, 2015

Legal Notices

Document id: p158-pd027601

Install Location Descriptors

The following names are used to define install locations throughout this guide.

Descriptor	**What the Descriptor Represents**
$ACTIVATOR_ETC	The install location of specific Service Activator files. The UNIX location is `/etc/opt/OV/ServiceActivator` The Windows location is `<`*`install drive`*`>:\HP\OpenView\ServiceActivator\etc`

In This Guide

This guide describes Service Activator's XMaps subsystem from the perspective of solution customization.

Audience

The audience for this guide is:

- Systems Integrator, who will use it as a resource for customizing solutions. The SI must be familiar with HPSA, HTML5 and Java programming for web pages, including Jacascript and JSP or JSF.

Other Documentation

This is one of several manuals for systems intergrators about HPSA. Familiarity with at least the following two manuals is assumed. Start with the Overview.

- *HP Service Activator, System Integrator's Overview*
- *HP Service Activator, Inventory Subsystem*

1 Introduction

XMaps is a component of HPSA. It allows the SI to define graphical diagram representations of networks, thus overcoming the limitation of HPSA Inventory to show inventory contents only as tree structures.

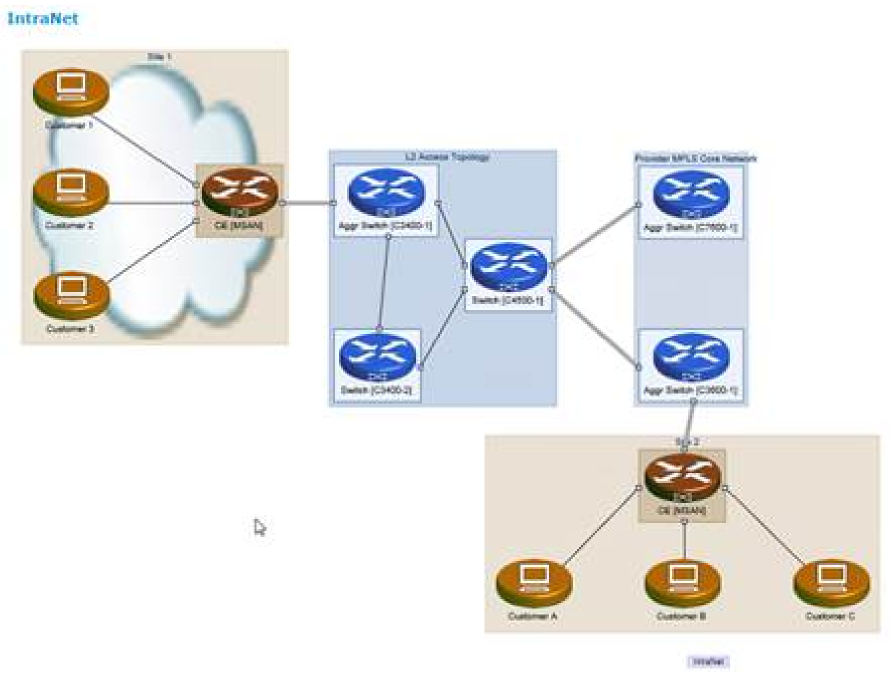
Objects that can be shown in XMaps diagrams are nodes, ports and connections. As a special case, object data can be specified statically in definition documents. Then the diagram appearance will be fixed, but generally objects will be retrieved by methods on specified Java bean classes that must be supplied by the integrator and object data can then be drawn from any source, allowing diagrams to represent a current object population.

XMaps is suitable for representing networks that can be modelled with the Common Network Resource Model (see *HP Service Activator, System Integrator’s Overview*) or models with similar objects to represent the network elements of older network architectures such as SDH and PDH. Actually nodes and connections can be used to show any (mathematical) graph, for example a state transition graph representing a fulfillment process, so the applicability of XMaps is even wider.

Diagram Structure

Figure 1-1 is an example of a network diagram as it can be shown by XMaps. You can imagine here that the brown devices on purple background belong to a customer intranet, with sites inter-connected via a virtual private network implemented over the provider network shown in blue. Nodes, ports (although the rendering is small) and connections are easily recognized in the diagram.

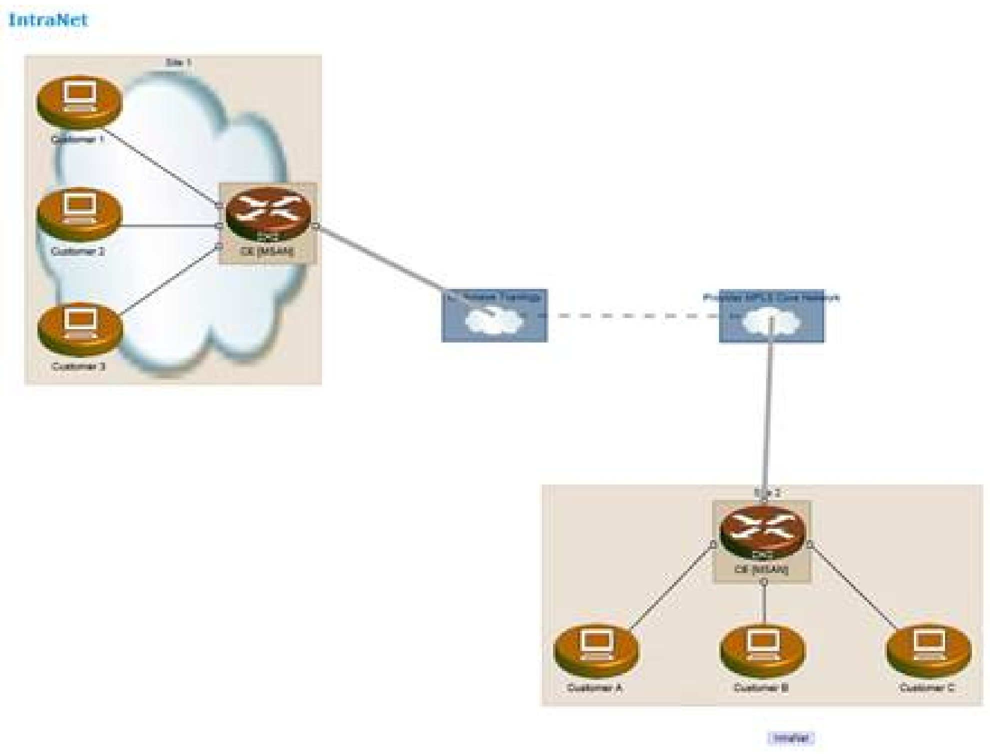
Figure 1-1 Example XMaps Diagram



The mentioned objects can be organized in groups to make up a complete diagram. In Figure 1-1 there are two groups, viz. the two sites of the intrenet owner. The example figure includes another feature of XMaps, which appears very similar to groups, but is subtly different, called embedded diagrams. Embedded diagrams are defined separately from the main diagram in which they appear. This allows them to be reused within different main diagrams. The embedded diagrams here are the blue ones showing the provider network. They could alternatively be shown embedded in the diagram for another customer’s intranet, or each of them could be shown on its own, as a main diagram, without a sorrounding context.

An embedded diagram can be collapsed to be rendered only as an icon, as shown in Figure 1-2. The same is true for groups.

Figure 1-2 Example Diagram with Embedded Diagrams Collapsed



XMaps allows the SI to control many aspects of diagram apperance: icons, colors, labels, connection lines, etc., as evident from the figures above. All of those aspects and how to control them are explained in chapter 2.

Diagrams can be incorporated in the HPSA UI, either in the working area of the main window, or associated with branches in the tree view of the inventory UI in the operations area (the area on the lower right where operation forms are shown by default). XMaps can be incorporated into the UI in one of two ways, either using one or more diagram definition documents and the XMaps taglib, or through a Java library that you can call from your own Java code (in a JSP or JSF). In either case XMaps will be processing a diagram request, and it will generate HTML5 code which is processed in an HTML session. Actually XMaps diagrams can be incorporated in any HTML frame, even outside of the context of an HPSA solution.

The API for the library you can invoke from your own Java code is documented as Javadoc that you can display when HPSA is installed; this chapter will serve as your introduction to the classes that occur in the library. The remainder of this manual is principally concerned with using diagram definitions, in which case generation of Java code is left to the XMaps taglib.

A diagram may have multiple layers, where each layer is again a diagram. This containment is not recursive: a layer cannot itself be layered. A layered diagram is defined by a list of references to the layer diagrams. When the layered diagram is displayed, the first (top) layer in the list is shown initially. The user can then click a layer launch button at the bottom of the diagram frame (the dock) to launch the diagram in the next layer. The idea is that each successive layer will show details at a deeper more detailed level. A node in the present layer must be selected to enable layer launching. Launching will generate a new request to process the diagram for the next layer. The request will include a list of layer parameters that are passed to the launched layer diagram as context information to allow that diagram to show desired information, for example inner details of the selected node. Conversely, it is always possible by clicking on a dock button to revert to the preceeding layer. In that case no context is selected: the higher layer will simply be shown again as it was left when the deeper layer was launched.

Diagram Definitions

XMaps diagram definitions are dedicated XML documents. Diagram definitions must be deployed in HPSA’s system database, before they can be processed by the XMaps taglib at runtime. XMaps includes a graphical tool, the XMaps Designer, which can be used to edit definition documents and deploy them. The main elements of XMaps definition documents define the visible objects (diagrams with layers and embedded diagrams, nodes, ports, connections and groups). They are complex elements composed of many smaller elements. At runtime the actual objects to be shown are typically retrieved by methods specified in the elements that define those objects, known as Find-By methods. Likewise, object definitions specify retrieved data contents (properties) as bean classes. It is straightforward to map methods and bean classes to HPSA inventory, in order to use XMaps to show inventory contents in diagram form.

Full details of diagram definitions are presented in chapter 2 and the XMaps Designer is described in chapter 3.

XMaps Objects

As mentioned above, the main visible object classes of XMaps are nodes, ports and connections. These object all have a number of predefined properties to which properties of associated beans can be added. The same is true for an entire diagram and for groups. The appearance of the objects in diagrams are controlled through th predefined properties.

The main characteristics of the object classes are briefly described here.

Node

Nodes are the fundamental objects of graphs and network. Simple networks are modelled flatly in a single layer, network elements are then modelled and shown as nodes.

In more complex cases, when networks are larger and must be broken down to allow meaningful display in a limited space, and also when it is interesting to look inside single network elements, networks can be broken down hierarchically and modelled with XMaps in multiple layers. Nodes will be the fundamental building blocks in all layers. A subnetwork may be represented as a node in a larger network. A physical network element may be represented as a node in a subnetwork. A part of a network element, such as a rack or a card may again be represented as a node in a more detailed layer. Except for ports, the XMaps node object does not model hierarchical containment; the model is flat, containment must be modelled by naming conventions (typically concatenation) using properties which can be used in Find-By methods to retrieve contained nodes.

Nodes can be pairwise interconnected, but this requires ports within the nodes to terminate the connections. If the ports are not explicitly defined, they will be hidden in the diagram, and the connection is shown terminating in the middle of each node.

Nodes can be grouped to organize a diagram. Nodes are displayed as icons, that can be colored and labelled.

Nodes have coordinate properties which are used to control their location within a diagram. The coordinates are optional. If they are not assigned, a placement algorith will determine where nodes are placed.

Port

Ports are modelled by XMaps as subobjects within nodes which serve as termination points for connections. Ports are important as an inner structure of network elements. They are displayed as small boxes on the perimeter of the nodes they belong to. A port can be defined as hidden, then it will have a place on the border of the node, where a connection can terminate, but no icon representing the port is shown; hidden ports cannot be selected.

Connection

Connections are between pairs of ports, hence between pairs of nodes. They can be used to represent physical cabling, communication paths, or in general arcs of graphs. They are displayed as lines, which can be colored, unbroken or dashed, and labelled.

Connections are not limited by groups; they can be inter-group as well as intra-group.

Group

Entire diagrams occupy rectangular areas within a displayed page. Groups are divisions wihtin a diagram, shown as non-overlapping rectangles. They can be used to represent subnetworks within a single flat diagram.

Each group is defined primarily by the nodes it contains. Groups have controllable display attributes, such as background color, border thickness and labels.

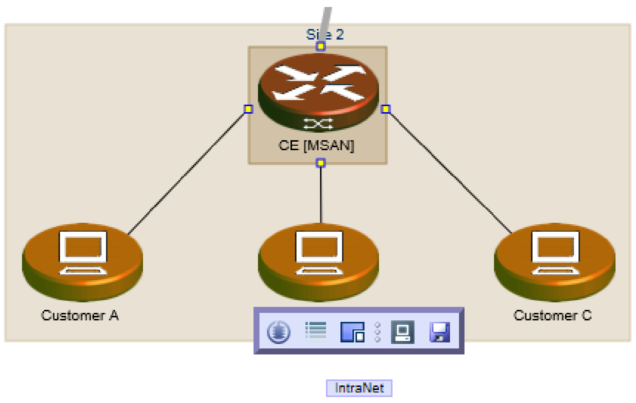
User Interactions

Users can interact with XMaps diagrams by selecting objects and dragging them or right-clicking to launch operations. Groups and embedded diagrams can be collapsed and expanded by double-clicking in the area representing the group/embedded diagram.

Selecting, dragging, collapsing and expanding all generate events for which XMaps can be configured, through the diagram definition, to perform actions. Likewise the definition of an object may include operations with associated actions which become selectable on right-click of a displayed object instance.

The bottom border of the diagram frame is called the dock. Dock operations, which will be launchable from icons on the dock, can be defined for a diagram. Launching of a deeper layer (described under Diagram Structure above) is also a dock operation.

Figure 1-3 **Diagram Dock**

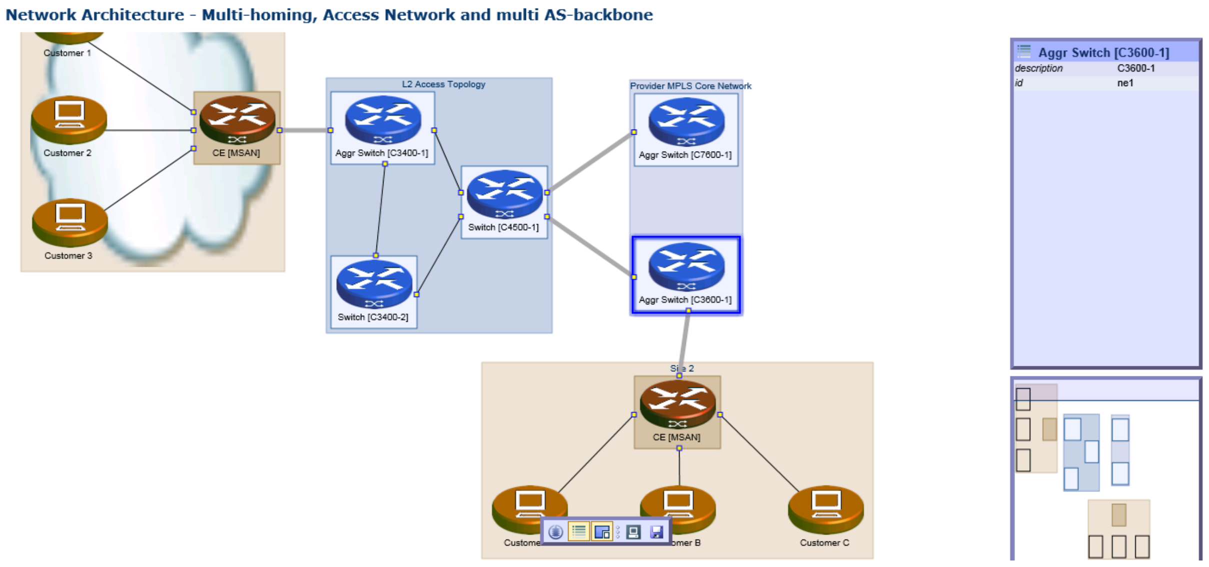


All actions associated with events and operations are executed as Javascript functions specified in the diagram definition. Javascript functions must be implemented by the SI. XMaps and HPSA Workflow Manager functionality can be invoked through their APIs.

Operations may be qualified by privileges. Privileges are assigned to users through roles as described in *HP Service Activator, System Integrator’s Overview*.

XMaps can show the properties of the diagram and of a selected object in a property box, as shown in Figure 1-3. Note in the figure the dock consisting of a row of five buttons. One of the buttons toggles on and off the display of the property box. If no object is selected the property box will show properties of the diagram.

Figure 1-4 **Diagram with Property Box**



Aggr Switch [C3600-1]

description	C3600-1
id	ne1

NOTE Below the property box is shown a navigation box. If the diagram is larger than can be shown in the frame of the screen window where it appears, this box will indicate which part of the complete diagram is currently visible.

Saving Coordinates

When a diagram is displayed based on a definition retrieved from HPSA’s system database the dock will include an option for saving the diagram coordinates. When this option is clicked the diagram coordinates will be saved in the database and will, upon reopening of the diagram, override any specified in the definition. If new elements are found in the diagram, their coordinates from the diagram definition will be used.

2 XMaps Definition Documents

XMaps definitions are XML documents which consist of elements and attributes with a syntax defined in a schema that you can study as file `$ACTIVATOR_ETC/config/xmaps.xsd` when HPSA is installed. In this section an equivalent plain English description is provided with explanations of the meaning of the elements and without the syntactic clutter of XML. The detailed information towards the end of the chapter is structured as tables which will be convenient for reference. If you are familiar with XML schema you may also like to use the schema as a reference. If not, don’t worry, this chapter contains all the information you need to understand the structure and meaning of XMaps definitions.

The visible objects to be defined are diagrams, layers, groups, nodes, ports and connections as introduced in Chapter 1. They can be statically defined, i.e. with all properties explicitly stated in the definition, but typically only some of the properties are specified in the definition, whilst remaining details are retrieved as beans. To allow such retrieval the definition element for each object must include the bean class and Find-By method(s) to use.

Bean properties can also be defined for the elements Background-Text and Background-Image. These elements are child elements of diagram elements and describe general properties of diagrams; unlike the proper objects background texts and images cannot be selected or dragged.

Diagram elements have a number of child elements which define general properties. Some of these are simple and have to do with the incorporation of the diagram into an overall HPSA solution, like name of the diagram, name of the HPSA solution it belongs to, names of privileges used for operations. Other simple child elements of diagrams are display properties, like colors and texts, and diagram behaviour properties such as whether nodes in the diagram can be dragged or selected. The layers, embedded diagrams, groups and nodes which make up the diagram are themselves defined by separate elements within the definition file; child elements of the diagram define its contents by references to these separate elements.

Each diagram, whether it is used stand-alone, as a layered diagram, as a single layer or an embedded diagram, must be separately defined and deployed in the system database with a solution name and a diagram name. References to deployed diagrams are by solution name and diagram name.

For complete tabular lists with explanations of all the elements and child elements, see the section titled Elements of XMaps Definitions below.

Operations and Events

Diagram, node, port and connection objects can have operations. Each operation has an action defined as a JavaScript function call, which can have any number of parameters specified by string values. The system integrator must also write (or otherwise obtain) the implementation of the called JavaScript functions and integrate them into the JSP or JSF that invokes XMaps.

Similarly, the actions to take by XMaps upon events generated by the user are defined as calls of Javascript functions. Events can be selection, dragging, or collapsing/expansion of embedded diagrams and groups. Elements of diagram definitions specifying event actions have names of the form On-x, like On-Select.

Object Types

If a diagram has many objects of the same class with similar properties, the property values can conveniently be defined as an object type (Group-Type, Node-Type, Port-Type, Connection-Type). Actual object are then defined with a reference to the object type. Individual properties defined for the actual object will overrule those of the object type.

Java Beans and Properties

Java beans may be defined for all objects and for Background-Text and Background-Image. The implementation of a Java bean must be defined with a Class element which references a deployed Java class. To determine which instances of the object shall be instantiated at runtime, a Find-By element must be provided alongside the Class. It defines the method and parameters (Keys) to call to retrieve bean instances. One object is instantiated for each bean instance that is retrieved. The Find-By element can generally be repeated. The first First-By element which is unconditional or whose Condition evaluates to *true* is then used.

The properties to be retrieved from each bean instance are defined with the Attributes element. The implementation of the bean must include getter methods for those properties.

All methods used to retrieve bean data, i.e. those mentioned in Find-By elements must take as their first parameter a datasource identifying the pool of database connections to use. The findBy methods generated by the inventory subsystem for resource beans satisfy this requirement.

In order to have an effect in XMaps, bean properties must be assigned to the predefined object fields with semantics described in the tables in the section below titled Elements of XMaps Definitions. This can be done using string value quoting (see below). In two special cases, for the elements Key and Node-Reference values can be fetched without converting them to strings. This is otherwise very similar to string value quoting, and the details are therefore given at the end of the section thus titled below.

Object Naming and Identification

The elements which define objects have an optional Name child element. Its value can be used in specific circumstances to make (cross-)reference to the element within the diagram definition. Hence the names must be unique across the definition document.

Beware that an element definition is typically instantiated as a number of object instances. Hence the name does not uniquely identify a runtime object. Nevertheless (but only for the object itself in whose definition it occurs, its encloser or the diagram) it is used for that purpose in references to bean properties, as described in the next section.

Runtime objects are uniquely identified by the value of the Id child element. The values are typically fetched from bean properties using string value quoting. Id values must be unique across all types of objects within a diagram at runtime. A useful technique to ensure uniqueness is to concatenate stored primary keys when they are only unique within one class of data, for example: node-id_port-id.

String Value Quoting

Values for simple elements and XML attributes are generally defined by strings. In such strings occurrences of various (sub-)string values can be incorporated by quoting: centrally named constants, attributes of the HTML session in which XMaps runs, parameters and attributes of the (taglib or other) request which is being processed by the session, and properties of a bean associated with an object, typically the object in whose definition the string occurs or the diagram. The $-character is used to open and end a quote.

Named constants are defined with names and values in the Case-Packet element.

Table 2-1 defines the syntax and explains the meaning for each of the forms a quoted string can take. Names in italics are used as placeholders for actual names of constants, attributes, parameters, attribute, properties or elements.

NOTE There is no general mechanism for cross-referencing values defined statically in other elements of the diagram as part of string values in definitions; for this purpose, define named constants and reuse them as necessary.

Table 2-1 Syntax for String Value Quoting

Quoting syntax	**Explanation**
$variable:*name*$	Refers to the value of the named constant
$variable.session:*name*$	Refers by name to an attribute of the HTM session
$variable.request.parameter:*name*$	Refers by name to a parameter of the servlet request being processed
$variable.request.attribute:*name*$	Refers by name to an attribute of the servlet request being processed
$*element*:*name*$	Refers by name to a property of a Java bean associated with an object or to a request attribute of the diagram. The object is named by *element*. Which objects can be referenced depends on the referring element; it is explained below.

Remember, all quoting occurs inside the definition of an object element (Diagram, Node, etc.). The *element* used to reference an object can be one of:

- the name defined (by Name child element) for the object for which a value is being defined; in a type definition (Node-Type, Port-Type, Connection-Type or Group-Type) just use `this`; use `this` also if the element has no name.

```
<Diagram>
  …
  <Node-Type name="router_type">
    <Image>$this.icon$</Image>
  </Node-Type>
  <Node>
    …
    <Node-Type-Reference>router_type</Node-Type-Reference>
    …
  </Node>
  …
</Diagram>
```

  When an element representation is later applied to an element definition, *this* will be used as the element name. According to this, the previous example will bring the same result as:

```
<Diagram>
  …
  <Node>
    <Name>router</Name>
    …
    <Image>$router.icon$</Image>
    …
  </Node>
  …
<Diagram>
```

- the name defined for the Diagram.

```
<Diagram>
  <Name>network</Name>
  <Class-Name>…</Class-Name>
  <Find-By>…</Find-By>
  …
  <Node>
    <Name>router</Name>
    <Id>$router.id$</Id>
    <Class-Name>…</Class-Name>
    <Find-By>…</Find-By>
    <Text>$network.label$ - $router.label$</Text>
    …
  </Node>
  …
</Diagram>
```

- in the case of a port object, the name defined for the node the port belongs to.

```
<Diagram>
  <Name>network</Name>
  <Class-Name>…</Class-Name>
  <Find-By>…</Find-By>
  …
  <Node>
    <Name>router</Name>
    <Class-Name>…</Class-Name>
    <Find-By>…</Find-By>
    …
    <Ports>
      <Port>
        <Name>interface</Name>
        <Id>$interface.id$</Id>
        <Class-Name>…</Class-Name>
        <Find-By>…</Find-By>
        <Text>$network.label$ - $router.label$ - $interface.label$</Text>
        …
      </Port>
    </Ports>
  </Node>
  …
</Diagram>
```

The same syntax as for string value quoting of a bean attribute, but without the quotes ($), can be used for the elements Node-Reference (in the definition of a group) and Key (in Find-By) to fetch any type of value, even multiple occurrences in an array and retain its type. Without the quote signs this syntax must be used as the entire string value; it cannot be concatenated. Multiple values will be retrieved to multiple occurrences of Node-Reference or Key. For Keys the types of the retrieved values must match those defined by the type attributes.

- Keys: The element <Key> is used to specify a parameter of a find-by method which can be of any type different than string. If a key value is quoted then it will be necessarily resolved to a string, otherwise the retrieved object (string or not) will be treated as the specified key type. As a consequence, quotes ($) are not needed in key values.

  The example below defines a string and a boolean parameters for the find-by method:

```
<Node>
  <Name>router</Node>
  <Id>…</Id>
  <Class-Name>com.hp.ov.activator.xmaps.test.Router</Class-Name>
  <Find-By>
    <Method>findByEnabledRouter</Method>
    <Key>diagram.id</Key>
    <Key type=”boolean”>router.enabled</Key>
  </Find-By>
  …
</Node>
```

- Node references in groups: As part of a group definition the nodes that are part of the group must be specified using the element <Node-Reference>. The value specified here can represent a single node identifier or an array of node identifiers. In the second case, the node reference cannot be enclosed by quote ($).

The example below contains both cases:

```
<Group>
  <Name>group</Name>
  <Id>…</Id>
  <Class-Name>com.hp.ov.activator.xmaps.test.Group</Class-Name>
  <Node-References>
    <Node-Reference>$group.mainNodeId$</Node-Reference>
    <Node-Reference>group.nodesIds</Node-Reference>
  </Node-References>
  …
</Group>
```

The field name is resolved following the next steps:

- First, a getter method is looked for. The method name must start with *get* followed by the field name where the first letter is replaced by the capital letter.
- If not found, a boolean getter is looked for. The method name must start with *is* followed by the field name where the first letter is replaced by the capital letter.
- If not found, a getter for an extended attribute is looked for. This is the only relationship between XMaps and Inventory Builder, this the method name generated by Inventory Builder will be looked for: *getExtendedAttibute*, which must receive a string as parameter so the name of the extended attribute can be specified through it.
- If not found, the value enclosed by character $ cannot be resolved and is returned as is.

Note that there is no limitation on the object type that can be returned by these getter methods, but take in mind that it will be the string representation of the object returned what will be displayed in the diagram. As a consequence, objects that doesn't have an understandable string representation are strongly not recommended.

Localization

XMaps in itself has no localization issues. If you define labels for diagrams, you may want to translate your diagram definition.

Elements of XMaps Definitions

An XMaps diagram definition will contain simple elements and successively more complex elements, composed of simpler elements which are included by embedding or by reference, and finally the most complex elements represent the main visible objects.

At runtime, every element instance will belong to an object; in the explanation of an element that object is referred to as simply "the object". To determine it, follow the chain of enclosing elements until an object element is reached.

NOTE We use capitalisation: Node, Port, etc., to refer to XML elements, but refer to objects and classes with lower-case initials: node, port, etc.

XML Attributes

A number of XML attributes are used in XMaps definitions. Some of them may occur on several elements. They are listed and their meanings explained in Table 2-2. All these attributes are string typed. The meaning may depend on the element that an attribute appears on.

NOTE The word attribute is used here to refer to the XML attributes of diagram definitions. The vocabulary of diagram definitions also contains Attributes/Attribute elements; these elements define (primarily) Java bean properties, and we use the term property to describe them. Let it not confuse you.

Table 2-2 XML Attributes of XMaps Definitions

Attribute name	**Mandatory / default value**	**Explanation**
alt	M	used on Dock-Operation; defines a tooltip for the operation
arrow	*false*	used on Origin-Id or Destination-Id of a connection to specify whether an arrow shall be shown at that end of the connection
color	black	used on Text element to define the color of the text, can be any valid HTML5 color specification
drag	*true*	on Diagram: if *true*, the diagram is permitted to contain draggable nodes on Node: if *true* and permitted for the diagram, the node will be draggable
dock	*true*	on Diagram; if *true*, the diagram will display the dock, otherwise the dock will never be displayed and dock operations will not be available
name	M	used in cross-references to objects to match the name of the defining element
role		used in Condition element, it is *true* if the user has the role given as value of the attribute
select	*true*	on Diagram: if *true*, the diagram is permitted to contain selectable objects on Port, Node or connection: if *true* and permitted for the diagram, the object will be selectable
solution		applies to Diagram-Reference and Embedded-Diagram; specifies the solution name used to look up the refernced diagram in the system database
strict	false	on Diagram: determines handling of multiple occurrences of identifiers in the definition; if *false*, object with second and further occurrences of an Id value will be ignored in a diagram; if *true*, multiple occurrences will cause diagram failure
toggle	false	applies to Dock-Operation, specifies if the operation toggles, i.e. switches between two states
toggled	false	the state of the toggling operation
type	String	used for a Key of Find-By, specifies its type: one of *string*, *int*, *long*, *boolean*, *double*, *float*, *date*

Simple Elements

Except for three cases simple element consists of only a simple value. The three exceptions are Condition, Key and Text which also have an attribute (identified in parentheses after the element name). The meaning of each simple element is explained in Table 2-3.

Simple elements are included as sub elements of more complex elements. All the values are string typed.

Table 2-3 Simple Elements of XMaps Definitions

Element name	**Explanation**
Action	applies to Operation; specifies as a Javascript function call the action to execute to perform the operation
Background	applies to diagram, embedded diagram, group, node or port: background color of the object, can be any valid HTML5 color specification
Border-Color	applies to diagram, group, node or port: border color of the object defined by the parent element; can be any valid HTML5 color specification
Border-Style	applies to diagram; specifies the style with which the border of the diagram is rendered, can be any valid HTML5 style specification, default is *inset*
Border-Width	applies to diagram and node: width of border of the object; values are specified as number of pixels
Box-Background-Color	applies to diagram; specifies the background color for the dock area and property box; can be any valid HTML5 color specification, default is light blue.
Box-Background-Color-Even	applies to diagram; specifies the background color for even rows of the property box, default is mid blue.
Box-Background-Color-Odd	applies to diagram; specifies the background color for odd rows of the property box, default is light blue.
Box-Background-Color-Title	applies to diagram; specifies the background color for the diagram title, default is light blue.
Box-Border-Color	applies to diagram; specifies the border color for the dock and the navigation and property boxes, default is dark blue.
Center-Ports	applies to node; if *true*, all ports on the node are hidden and placed in the center of the node
Class-Name	applies to all objects that can have associated beans; specifies full name of the class that implements the bean
Collapsed	applies to group and embedded diagram; boolean expression whose value determines whether the object must be collapsed initially when its parent diagram is displayed
Collapsed-Background	applies to group and embedded diagram; specifies its background color when it is collapsed, can be any valid HTML5 color specification
Color	applies to connection: specified the color with which it is displayed, can be any valid HTML5 color specification
Condition (role)	boolean expression; role can be used in the expression if the parent element is Operation, *false* will disable the operation, then it will not be shown in the menu otherwise when the parent element (for example Find-By) is repeated as a member of the grandparent, only the first parent with a *true* condition will be used

Element name	**Explanation**
Dash	applies to connection; if this element is present, the connection will be shown as a sequence of dashes with length in pixels given by the value, otherwise as an unbroken line
Datasource	applies to diagram; names the datasource to use to access beans in database
Description	applies to Operation-Type-Name or Diagram; a textual description to be shown in editor tools
Diagram-Name	applies to diagram; specifies the name with which a diagram definition is deployed in the system database (together with the solution name) and referenced for launching into XMaps for viewing
Font	applies to Background-Text; defines the font for the text, can be any valid HTML5 font definition; default is arial
From	belongs to Attribute-Mapping; defines name of a layer parameter in the launching layer
Front-Image	applies to group and embedded diagram; specifies by file name an image to be shown for the object when it is expanded
Height	applies to node object; specifies as number of pixels the height of the node, can be overridden when a taller label or image is specified
Hidden	applies to port object; if *true*, the port is not shown; default is *false*
Id	mandatory on all objects; value must be unique among all objects in a diagram at run-time; serves internal technical purpose
Image	applies to Dock-Operation, node, connection, Background-Image, group, embedded diagram; specifies by file name an image to be shown for the object; for Dock-Operation the image is shown on the dock for group and embedded diagram the image is shown only when the object is collapsed for connection the image is shown in the middle of the connection
Key (type)	specifies a parameter for Find-By method by type and value; key order must match parameter order on the method; the first is always the datasource which is not defined by a key if Key values are stated literally they will converted to the type; see the section String Value Quoting above for a description of how to retrieve key values from beans
Line-Type	applies to Connection; specifies how must the connection be represented. Accepts the values *straight* (default), *double_elbow*, *double_elbow_horizontal*, *double_elbow_vertical*, *double_elbow_top*, *double_elbow_left*, *double_elbow_bottom*, *double_elbow_right*, *elbow_horizontal* and *elbow_vertical*

Element name	**Explanation**
Method	applies to Find-By; specifies the method to call in the bean class (specified by Class element) to retrieve beans to instantiate an object element
Name	names the containing element to enable cross-referencing within the definition document
Node-Reference	used in definition of a group to refer to a node to include it as a member of the group; the value must equal the Id of the node see the section String Value Quoting above for a description of how to retrieve multiple node ids as a single bean property
On-Select	applies to port, node, connection and diagram; the value is a JavaScript function call that is executed when the object is selected, on diagram when any object in the diagram is selected
On-Unselect	applies to port, node, connection and diagram; the value is a JavaScript function call that is executed when the object is unselected, on diagram when any object is unselected
On-Collapse	applies to group and diagram; the value is a JavaScript function call that is executed when the group (for diagram: any group) is collapsed
On-Expand	applies to group and diagram; the value is a JavaScript function call that is executed when the group (for diagram: any group) is collapsed
On-Select-Multiple-Start	applies to diagram; the value is a JavaScript function call that is executed when selection of multiple object starts
On-Select-Multiple-End	applies to diagram; the value is a JavaScript function call that is executed when selection of multiple object completes
On-Drag-Start	applies to diagram; the value is a JavaScript function call that is executed when dragging of an object starts
On-Drag-End	applies to diagram; the value is a JavaScript function call that is executed when dragging of an object ends
Operation-Type	element of Operation or Dock-Operation; the value refers to a privilege (defined by Operation-Type-Name), which the user must have for the Operation to be available
Script	applies to diagram; full name of a Javascript file that will be included in JSP controlling the diagram display; Javascript functions invoked by actions can be implemented in such files
Solution	applies to diagram; specifies the solution name with which the diagram definition is deployed in the system database
Sorting-Level	applies to nodes; used by sorting algorithm, must be positive integer
Text (color)	applies to port, node, connection, group and Background-Text; the value is a text string which is shown as object label or background text

Element name	Explanation
Title	applies to diagram; specifies a diagram title, shown at the bottom of the diagram
To	belongs to Attribute-Mapping; defines name of a layer parameter in the launched layer
Tooltip	applies to port, node and connection; the value is a text that is shown as a tooltip when the cursor hovers over the object
Value	used in Attribute and Variable-Value to specify the value for a bean property, layer parameter or named constant; the value is specified as a string which may include quoted string values
Width	applies to node and connection; specifies as number of pixels the width of the object, for node it can be overridden when a wider label or image is specified
X	applies to node, Background-Text, Background-Image and embedded diagram; specifies x coordinate of object location in the diagram measured in pixels
Y	applies to node, Background-Text, Background-Image and embedded diagram; specifies y coordinate of object location in the diagram measured in pixels

Complex Elements

Complex elements are defined by sequences of member elements and attributes. Member elements can be simple (defined above) or other (less) complex elements. Attribute lists are preceeded by A.

For each member in the list, O means the member can be omitted, M that it is mandatory, R that it can be repeated.

Repeatable members may represent alternatives, where the selection is made at runtime. Such members will contain a Condition element. The Conditions are evaluated as boolean expressions on each repeated member until a true value is reached, then that member is selected.

Table 2-4 Complex Elements of XMaps Definitions

Element name	Members	Explanation
Attribute	Name M, Value M	member of a list in an Attributes or Layer-Attributes element; encloses a definition by name and value of a bean property or layer launch parameter for the parent object (sorry, the use of the element name Attribute is misleading)
Attributes	Attribute R	applies to port, node, connection and diagram; encloses a list of Attribute elements which define properties for the bean associated with the object

Element name	**Members**	**Explanation**
Attribute-Mapping	From M, To M	part of Diagram-Reference, i.e. definition of layered diagram; defines mapping of layer parameter name from the one used in the launching (From) diagram (the one referenced by the occurrence of Diagram-Reference in which it appear) to the one used in the launched diagram (To).
Connection-Type	Text O, Color O, Width O, Image O, Dash O, Tooltip O, A: name M	names and defines a collection of property values which can be used as base for definition of an actual Connection
Connection-Type-Reference	Condition M A: name	applies to Connection; the name attribute references a Connectiont-Type element, the Connection will have the properties defined by that element;
Destination-Id	A: arrow O	applies to connection; the value is the Id of the destination port or node
Diagram-Reference	Attribute-Mapping R, A: name M, solution	applies to layered diagram; identifies it by diagram and solution name for retrieval from system database
Dock-Operation	Image M, Condition O, Operation-Type O, Action O A: alt, toggle, toggled	applies to diagram; specifires a dock operation; if the condition is *false*, the operation will not appear
Dock-Operations	Dock-Operation R	applies to diagram, list of custom dock operations
Find-By	Condition O, Method M, Key R	applies to all objects associated with beans; each Find-By element defines a method with parameters to retrieve actual bean objects to show in diagram
Group-Type	Text O, Background O, Collapsed-Background O, Front-Image O, Image O, Collapsed O A name M	names and defines a collection of property values which can be used as base for definition of an actual Group
Group-Type-Reference	Condition M A name	applies to Group; the name attribute references a Groupt-Type element, the Group will have the properties defined by that element;
Initial-Case-Packet	Variable-Value R	applies to diagram; defines a list of named constant values
Layer-Attributes	Attribute R	applies to node; defines – by the member list – layer launch parameters which are passed to the diagram in the next layer when a transition to that layer is requested while the node is selected

Element name	Members	Explanation
Node-References	Node-Reference R	used in group definition to specify by a list of references which nodes (from those making up the parent diagram) belong to the group; there must be at least one
Node-Type	Text O, Image O, Background O, Border-Color O, Border-Width O, Center-Ports O, Sorting-Level O, Tooltip O, X O, Y O, Width O, Height O A name M	names and defines a collection of property values which can be used as base for definition of an actual Node
Node-Type-Reference	Condition O A name	applies to node; the name attribute references a Node-Type element, the node will have the properties defined by that element;
Operation	Name M, Condition O, Operation-Type O, Action M	occurs as a member of an operation list (Operations), specifies one operation; at runtime the operation will only be shown in the list if its condition (if present) is *true*
Operations	Operation R	applies to node, port, connection or diagram; contains list of operations that are available in right-click menu for the object
Operation-Type-Name	Name M, Description O	applies to diagram; defines and names a privilege for XMaps operations
Origin-Id	A arrow O	applies to connection; the value is the Id of the origin port or node
Port-Type	Text O, Background O, Border-Color O, Hidden O, Tooltip O A name M	names and defines a collection of property values which can be used as base for definition of an actual Port
Port-Type-Reference	Condition O A name	applies to Port; the name attribute references a Port-Type element, the Port will have the properties defined by that element;
Variable-Value	Name M, Value M	used in Initial-Case-Packet; specifies name and value for a named constant

Elements for Diagram Objects

The elements described here are those for which the corresponding objects can be retrieved as beans using Find-By methods. When Find-By methods are used, a single element in the definition may result in multiple objects. When Find-By methods are not used, each element represents a single object all of whose properties must be defined explicitly.

Background-Text and Background-Image are properties of diagrams. They are included in this section because thay can be retrievable as beans even though they are not proper objects.

Ports, nodes, connections and groups may be partly defined, i.e. with a subset of their properties, by X-Type elements, which can be referenced by name (X-Type-Reference), listed in Table 2-4 in the section above.

A diagram may be final, i.e. a single diagram, or it may be layered, consisting of a list of final diagrams. In the latter case it is defined by a list of references (a Layers element) to the final diagrams, which must be separately defined and deployed. Accordingly the Diagram element takes two different forms which are listed separately.

Table 2-5 Elements for Visible XMaps Objects

Element name	**Members**	**Explanation**
Background-Text	Name O, Id M, Condition O, Class-Name O, Find-By R, Text M, Font O, X O, Y O	applies to diagram; specifies a text to be shown in the background of the diagram
Background-Image	Name O, Id M, Condition O, Class-Name O, Find-By R, Image M, X O, Y O	applies to diagram; specifies an image to be shown in the background of the diagram
Port	Name O, Id M, Condition O, Class-Name O, Find-By R, Port-Type-Reference R, Text O, Background O, Border-Color O, Hidden O, Tooltip O, Attributes O, Operations O, On-Select O, On-Unselect O A: drag O, select O	defines one or more port objects with properties; if no Find-By is present a single port is statically defined; some of the properties can optionally be specified by references to Port-Type elements
Ports	Port R	element of Node, encloses a list of the Port elements defining the ports that belong to the node
Node	Name O, Id M, Condition O, Class-Name O, Find-By R, Node-Type-Reference R, Text O, Image O, Background O, Border-Color O, Border-Width O, Center-Ports O, Sorting-Level O, Tooltip O, X O, Y O, Width O, Height O, Layer-Attributes O, Attributes O, Operations O, On-Select O, On-Unselect O, Ports O A: drag O, select O	defines one or more node objects with properties; if no Find-By is present a single node is statically defined; some of the properties can optionally be specified by references to Node-Type elements
Connection	Name O, Id M, Condition O, Class-Name O, Find-By R, Connection-Type-Reference R, Text O, Color O, Width O, Image O, Dash O, Tooltip O, Origin-Id M, Destination-Id M, Attributes O, Operations O, On-Select O, On-Unselect O	defines one or more connection objects with properties; if no Find-By is present a single connection is statically defined; some of the properties can optionally be specified by references to Connection-Type elements
Group	Name O, Id M, Condition O, Class-Name O, Find-By R, Group-Type-Reference R, Text O, Background O, Collapsed-Background O, Front-Image O, Image O, Collapsed O, Node-References M, On-Collapse O, On-Expand O	defines one or more group objects with properties; if no Find-By is present a single group is statically defined; some of the properties can optionally be specified by references to Group-Type elements

Element name	**Members**	**Explanation**
Layers	Diagram-Reference R	applies to layered diagram; encloser for list of references to its layer diagrams
Embedded-Diagram	Id M, Background M, Collapsed-Background M, Front-Image M, Image M, Collapsed M, X M, Y O, Attributes	specifies that a separately defined diagram shall be embedded in the one where this element occurs
Diagram	Diagram-Name O, Solution O, Description O, Layers O	this form defines a diagram as consisting of layers
Diagram	Diagram-Name O, Solution O, Description O, Name O, Id M, Datasource O, Class-Name O, Find-By R, Title O, Background O, Border-Color O, Border-Width O, Border-Style O, Box-Background-Color O, Box-Background-Color-Title O, Box-Background-Color-Odd O, Box-Background-Color-Even O, Box-Border-Color O, Script R, Operation-Type-Name R, Attributes O, Operations O, Dock-Operations O, On-Select O, On-Unselect O, On-Collapse O, On-Expand O, On-Select-Muliple-Start O, On-Select-Muliple-End O, On-Drag-Start O, On-Drag-End O, Node-Type R, Port-Type R, Connection-Type R, Group-Type R, Embedded-Diagram R, Node R, Connection R, Background-Text R, Background-Image R, Group R, Initial-Case-Packet O A: drag O, select O, strict O	this form defines a final diagram

3 XMaps Designer Tool

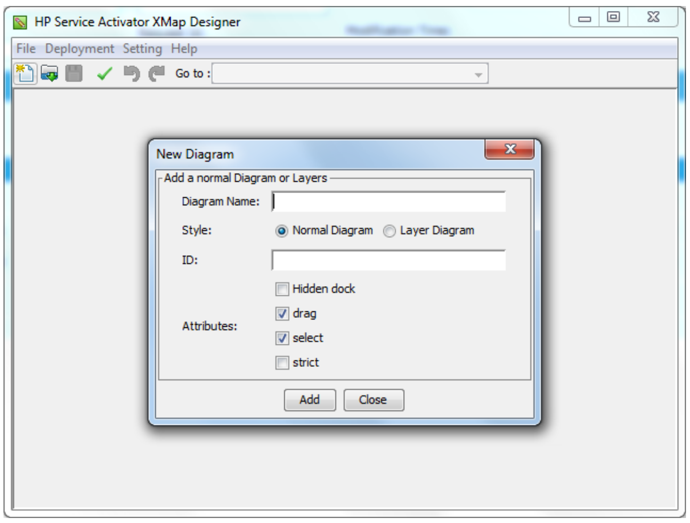
The XMaps Designer is a stand-alone tool in the family of HP Service Activator design time tools. It is basically a local Windows-based UI tool for editing diagram definitions with additional capabilities to validate and deploy them. All the functions except editing are also available as command line functions.

As the tool is quite simple, it has no dedicated manual; it is fully described in this chapter.

This chapter does not explain the contents of diagram definitions, i.e. the objects that are defined by elements of a diagram definition and their properties. Read chapter 2 before you start and refer back to it as a reference for details.

Figure 3-1 shows the user interface of the XMaps Designer, where the first step for creating a new diagram definition is open. The tool is asking for a diagram name, for the kind of diagram to be defined, either final or layered, and for the global attributes of the diagram.

Figure 3-1 XMaps Designer, Diagram Name and Kind


HP Service Activator XMap Designer
File Deployment Setting Help
Go to :
New Diagram
Add
Close

Add a normal Diagram or Layers

Diagram Name:

Style: [✓ Normal Diagram] Layer Diagram

ID:

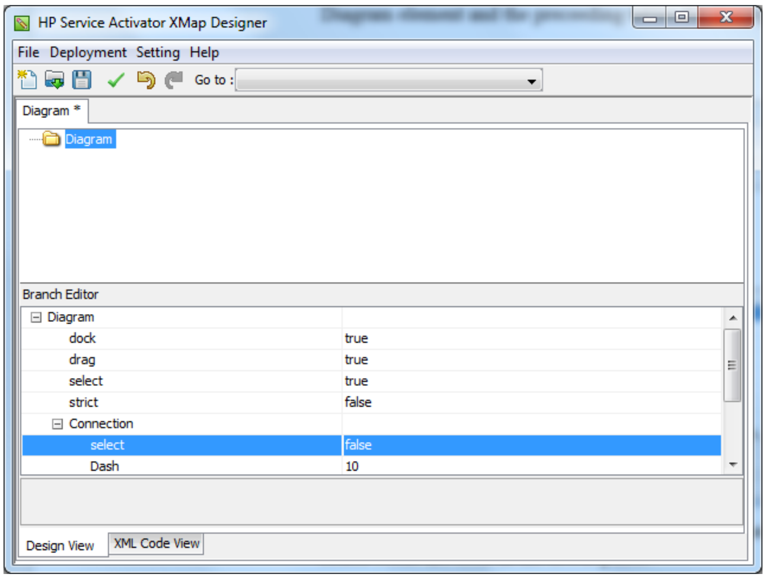
Attributes:
- ☐ Hidden dock
- ☑ drag
- ☑ select
- ☐ strict

Once you have created a diagram the XMaps Designer will have two panels: the top one showing the Diagram and its Node child elements in a simple tree structure, and the lower one showing child elements of the element which is selected in the tree as well as properties (attributes and simple elements). Node-Types, Groups, Group-Types, Connections, Connection-Types, Background-Texts and Background-Images are all children of the Diagram and shown in the lower panel when the root node is selected in the upper panel. Ports and Port-Types are children of Nodes and shown in the lower panel when the parent Port is selected in the upper panel. Properties of elements shown in the lower panel are indented under the line identifying the element as a child of the Diagram or a Node.

Right-click on a branch, in either panel, to get a menu of available element editing edit operations: copy/delete the element as a whole or add/paste a child element. To add optional properties to an element, right-click on its branch line in the lower panel, and select the property to add from the pop-up menu. To edit property values, select from drop-down menus when available, or edit the string values. To delete an optional property, right-click on it and select the delete operation.

In Figure 3-2 a final diagram has been created. There are no object child elements yet. The Diagram element branch is selected in the lower panel, allowing the user to perform edit operations on the Diagram element as a whole. Refer to Table 2-5 for the definition of the Diagram element and the preceeding tables for descriptions of the simpler child elements.

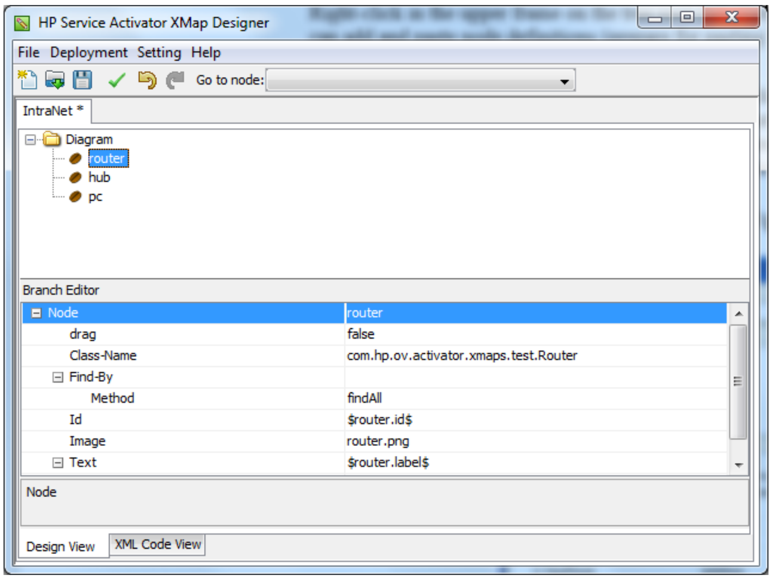
Figure 3-2 XMaps Designer, Diagram Properties


HP Service Activator XMap Designer
File Deployment Setting Help
Go to :
Diagram *
Diagram
Branch Editor
Design View
XML Code View

⊟ Diagram	
dock	true
drag	true
select	true
strict	false
⊟ Connection	
select	false
Dash	10

In Figure 3-3 three Nodes have been added: router (selected), hub and pc, and properties of the router Node are seen in the lower panel.

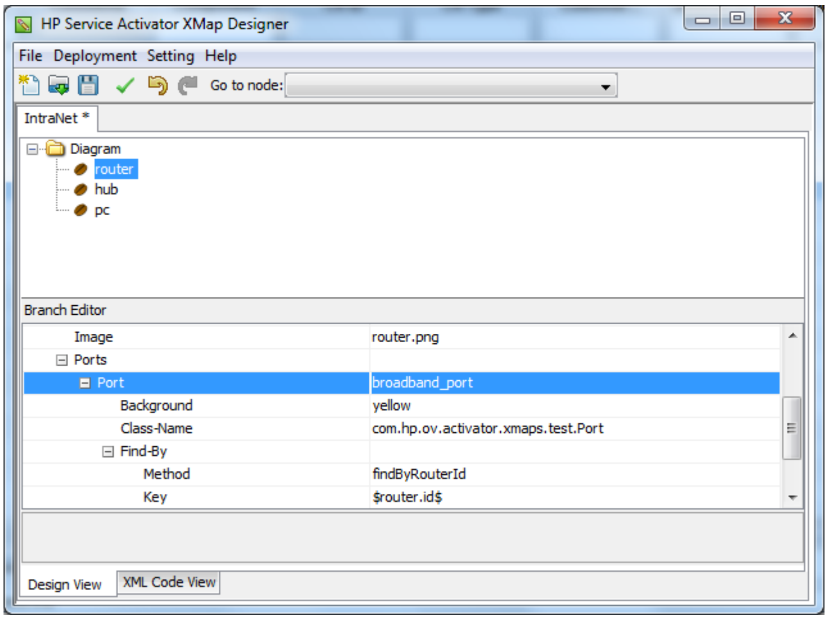
Figure 3-3 XMaps Designer, Node Properties and Parameters



⊟ Node	router
drag	false
Class-Name	com.hp.ov.activator.xmaps.test.Router
⊟ Find-By	
Method	findAll
Id	$router.id$
Image	router.png
⊟ Text	$router.label$

In Figure 3-4 a Port (named broadband_port) has been added to the router Node, and the Port properties are shown.

Figure 3-4 **XMaps Designer, Port Properties**


HP Service Activator XMap Designer
File Deployment Setting Help
Go to node:
IntraNet *
Branch Editor
Design View
XML Code View

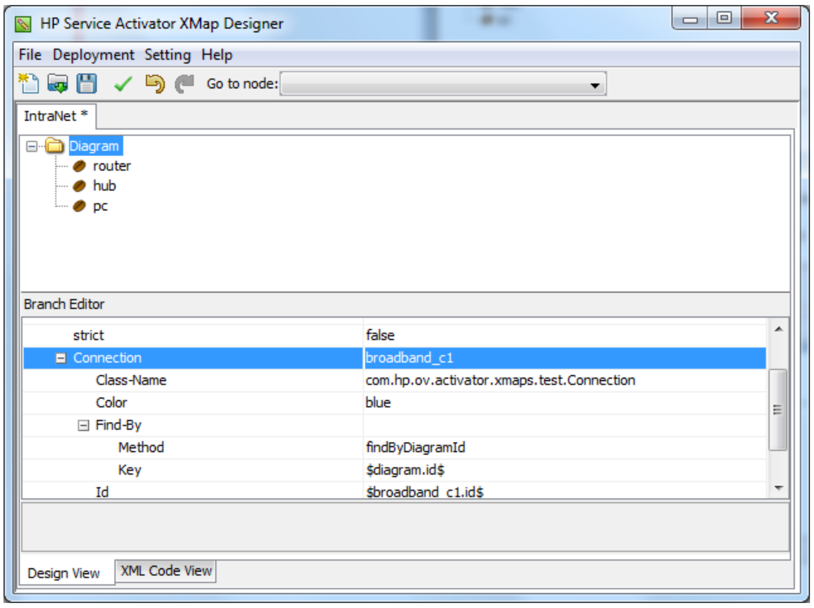
```
Diagram
  router
  hub
  pc
```

Image	router.png
Ports	
Port	broadband_port
Background	yellow
Class-Name	com.hp.ov.activator.xmaps.test.Port
Find-By	
Method	findByRouterId
Key	$router.id$

In Figure 3-5 a Connection has been added as a child of the Diagram. Remember the endpoints of the Connection must be defined by the mandatory properties Origin-Id and Destination-Id. Scrolling is needing to show them.

The other child elements of Diagram including Group appear in ways similar to Connection.

Figure 3-5 **XMaps Designer, Connection Properties**

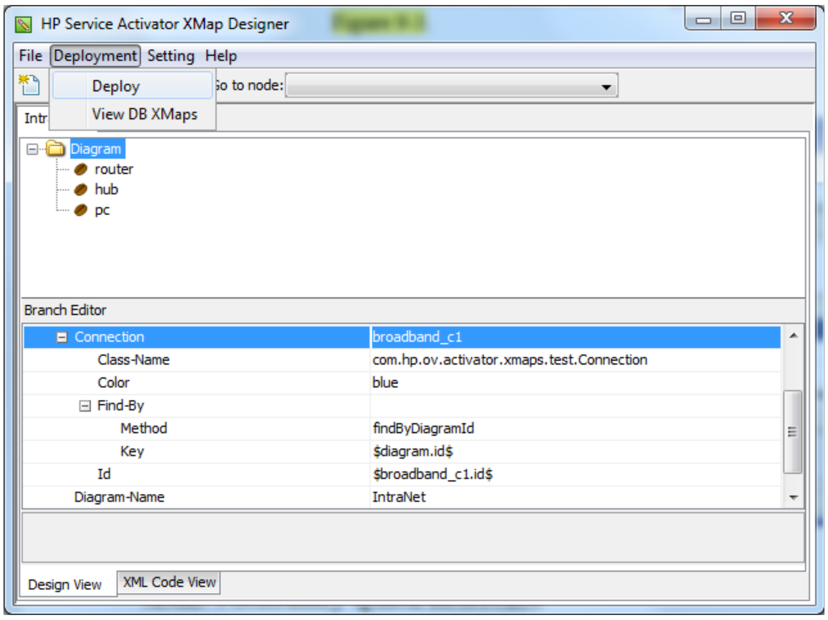

HP Service Activator XMap Designer
File Deployment Setting Help
Go to node:
IntraNet *
Branch Editor
Design View
XML Code View

```
Diagram
  router
  hub
  pc
```

strict	false
Connection	broadband_c1
Class-Name	com.hp.ov.activator.xmaps.test.Connection
Color	blue
Find-By	
Method	findByDiagramId
Key	$diagram.id$
Id	$broadband_c1.id$

When you have finished defining a diagram, including the properties Diagram-Name and Solution, you can deploy it in the database so it is ready to be used in Service Activator, possibly referenced from another diagram definition. Select the option *Deploy* from the *Deployment* menu as shown in Figure 3-6.

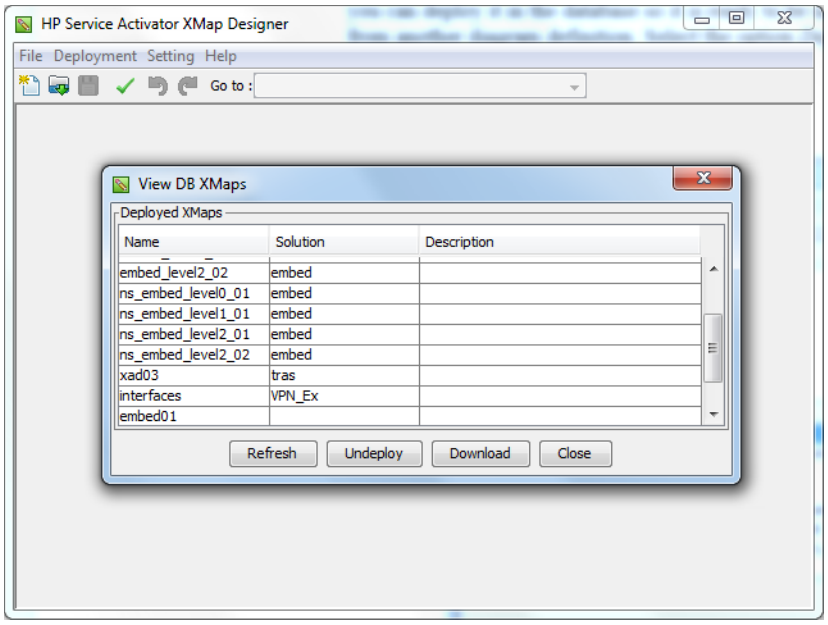
Figure 3-6 XMaps Designer Deployment Menu



Connection	broadband_c1
Class-Name	com.hp.ov.activator.xmaps.test.Connection
Color	blue
Find-By	
Method	findByDiagramId
Key	$diagram.id$
Id	$broadband_c1.id$
Diagram-Name	IntraNet

In case you want to see a list with the diagram definitions that are currently deployed in Service Activator select the option *View DB XMaps* from the *Deployment* menu. This option can be seen in Figure 3-6. When the list is displayed you can undeploy a diagram definition or download its contents to a local file as shown in Figure 3-7.

Figure 3-7 XMaps Designer View DB XMaps Menu



Name	Solution	Description
embed_level2_02	embed	
ns_embed_level0_01	embed	
ns_embed_level1_01	embed	
ns_embed_level2_01	embed	
ns_embed_level2_02	embed	
xad03	tras	
interfaces	VPN_Ex	
embed01		

Usage from Command Line

This chapter describes how to use the XMaps Deployer from a command line to process diagram definition files and deploy them to a running system.

The UI may build its run-time representation of the diagrams that are displayed in client browsers from a database table representation of the diagram definitions in Service Activator’s static repository. The run-time representation is updated from the repository every time it is requested, so it is not necessary to restart Service Activator after deploying one or more diagram definitions.

The command line syntax is:

```
XMapDesigner <function> <DB options> <other options> <diagram file>*
```

The <function> part identifies the requested function and function-specific options. The other parts are similar for all functions. The common parts are described first, followed by a small section for each function.

NOTE Options are not case sensitive. Values containing space(s) must be surrounded by double-quotes.

Table 3-1 **DB Options**

Option	Description
-dbHost <DBHOST>	optional, name of the database host
-dbName <DBSID>	optional, name of the database instance
-dbPort <DBPORT>	optional, port where database is accessed
-dbUser <DBUSER>	mandatory, name of the database user
-dbPassword <DBPASSWORD>	mandatory, password for the database user

DB optional options can get values from the system configuration files.

Table 3-2 **Other Options**

Option	Description
-verbose	causes XMaps Deployer to output verbose progress information including stack trace if an error occurs
-help	outputs list of valid options and syntax information
-version	outputs the version of the product

Deploy Diagram Definitions

In the command to deploy one or more diagrams the <function> part has this form:

```
-deploy
```

It is possible to specify the name of a single diagram definition file or the name of a directory; in the latter case all files with .xml suffix in the directory are processed as diagram definitions. The file names are not stored in the repository. Diagram definitions are identified by the diagram name and, optionally, solution name that are defined by the respective elements of the diagram definition.

List Diagram Definitions

In the command to list diagrams found in the repository the <function> part has this form:

```
-list
```

No names of diagram definition files can be entered for this function.

Delete Diagram Definition

In the command to delete a diagram definitions from the repository the <function> part has this form:

```
-delete [-force] [-solution <SOLUTION>] –diagramName <DIAGRAMNAME>
```

No names of diagram definition files can be entered for this function.

The -force option must be present if the specified diagram contains definitions of privileges for operations.

The <SOLUTION> and <DIAGRAMNAME> values are the names whereby the diagram is identified in the repository. They originate from the respective elements of the diagram definition.

Extract Diagram Definition

In the command to extract a diagram definition from the repository to XML format the <function> part has this form:

```
-download [-solution <SOLUTION>] –diagramName <DIAGRAMNAME> <TARGETFILE>
```

One file name must be given to name the target file. The options identify the diagram definition in the repository in the same way as for -delete.

4 Integrating XMaps in a Web UI

HTML Integration

When the diagram definition is finished it is ready to be displayed. Service Activator provides a tag library that can be used when working with Java Server Pages (JSP) and a facelet to use with Java Server Faces (JSF). Both tools generate the same source code (strict HTML 5) to display the diagram inside the resulting webpage.

To use the tag library in a JSP you just need to add a reference like the one below:

```
<%@ taglib uri="/WEB-INF/xmaps-taglib.tld" prefix="xmaps" %>
```

To use the facelet in a JSF you just need to defin the facelet prefix like the one below:

```
<html xmlns="http://www.w3c.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:xmaps="http://www.hp.com/xmaps/facelets">
```

In both cases, two tags will be available:

<xmaps:draw>	Displays the diagram.
<xmaps:update>	Replaces the previous diagram with another one. This tag can be used to switch to a completely different diagram or to redraw the first one with a number of changes.

NOTE The generated source code for displaying a diagram is strict HTML 5 and uses features that are not available in previous releases of HTML, and this is why it is important to use the browsers officially supported by Service Activator.

The example below belongs to a JSP which is displaying a diagram whose definition must have been deployed in the database with the diagram name *IntraNet* and solution name *example*.

```
<!DOCTYPE html>
<%@ page pageEncoding="utf-8"%>
<%@ taglib uri="/WEB-INF/xmaps-taglib.tld" prefix="xmaps" %>
<html>
<head>
  <title>IntraNet</title>
</head>
<body>
  <xmaps:draw scope="database" name="IntraNet" solution="example" />
</body>
</html>
```

In the example the `<xmaps:draw>` tag has three attributes: scope, name and solution. The full set of supported attributes is described in the following list:

- scope: Specifies from where the diagram definition is going to be obtained. It is a required attribute. Valid values are:

  request: The diagram must be retrieved as a request attribute. The name of the request attribute is specified by the attribute *id*. In this case it is assumed that the diagram is already resolved as it is the Java representation what is going to be obtained instead of the XML definition.

session: The diagram must be retrieved from a session attribute. The name of the session attribute is specified by the attribute *id*. In this case it is assumed that the diagram is already resolved as it is the Java representation what is going to be obtained instead of the XML definition.

session_manager: The diagram must be retrieved from a *Map* stored in the user session. This *Map* is internally managed by XMaps, and the key with which the diagram definition must be specified by the attribute *session_key*. In this case it is assumed that the diagram is already resolved as it is the Java representation what is going to be obtained instead of the XML definition.

file: The diagram definition must be retrieved from a file, which is specified by the attribute *file*.

database: The diagram definition must be retrieved from the database. It will be identified by the attributes *name* and *solution*.

body: The diagram definition must be retrieved from the contents of this tag. This option is not available in the facelet due to restrictions related to JSF environment.

- identifier: Specifies the name of the session or request attribute containing the diagram definition. It is required when the *scope* is *session* or *request*. When working with JSF instead of JSP
- file: Specifies the file name containing the diagram definition. It is required when the *scope* is *file*.
- name: Specifies the diagram name with which the diagram definition must be retrieved from the database. Thus, it must match the value of the element <Diagram-Name> in the diagram definition. It is required when the *scope* is *database*.
- solution: Specifies the solution name of the diagram definition. Thus, it must match the value of the element <Solution> in the diagram definition. It is required when the *scope* is *database* and the solution name is not empty.
- sort: Specifies whether the diagram has to be sorted or not, and in such case which algorithm must be used. Valid values are:

  sugiyama: the diagram nodes will be categorized in horizontal levels and sorted in a way that minimizes the number of crosses between connections.

  ring: similar to *sugiyama* but the nodes are displayed in concentric rings.
- gap: Specifies a custom separation between sorted nodes, in pixels. It is an optional attribute that is only relevant if a sorting algorithm has been configured.
- animate: Boolean attribute that specifies whether the diagram nodes have to be automatically dragged from the upper left corner to their respective coordinates or not. Defaults to *false* (no animation).
- store_in_session: Specifies whether the resulting diagram, once resolved and translated to Java objects, must be stored in the user session or not. Defaults to *false*, this meaning that it won't be stored. Storing a diagram in the user session may be useful if it is later going to be modified as part of the business logic of the solution and then updated.
- session_key: Specifies the name of the session attribute where the resulting diagram has to be stored. It is required when *store_in_session* is *true*. It also specifies the key name of the *Map* stored in the user session from where the diagram must be obtained; in this case it is required when the *scope* is *session_manager*.

The tag <xmaps:update> accepts the following attributes:

- scope: Same as the one for <xmaps:draw>.
- id: Same as the one for <xmaps:draw>.

- file: Same as the one for <xmaps:draw>.
- name: Same as the one for <xmaps:draw>.
- solution: Same as the one for <xmaps:draw>.
- sort: Same as the one for <xmaps:draw>.
- gap: Same as the one for <xmaps:draw>.
- animate: Boolean attribute that specifies whether the diagram nodes that already existed in the previous diagram must be automatically dragged to their new coordinates (those that didn't exist will be dragged from the upper left corner) or not. Defaults to *false* (no animation, nodes will just appear in their coordinates).
- store_in_session: Same as the one for <xmaps:draw>.
- session_key: Same as the one for <xmaps:draw>.

Example JSP

The example below belongs to a JSP which displays a diagram. It is assuming that the diagram definition contains an operation which calls the JavaScript function *update()*. This function will open a JSP in a hidden iframe.

```
<!DOCTYPE html>
<%@ page pageEncoding="utf-8"%>
<%@ taglib uri="/WEB-INF/xmaps-taglib.tld" prefix="xmaps" %>
<html>
<head>
  <title>IntraNet</title>
  <script>
  function update() {
    var url = "/activator/jsp/xmaps/update.jsp?id=myxd";
    window.open(url, "upd");
  }
  </script>
</head>
<body>
  <xmaps:draw scope="database" name="IntraNet" solution="example" />
  <iframe name="upd" style="display:none;"></iframe>
</body>
</html>
```

The JSP in the hidden iframe will update the diagram displayed in the parent JSP:

```
<!DOCTYPE html>
<%@ page pageEncoding="utf-8"%>
<%@ taglib uri="/WEB-INF/xmaps-taglib.tld" prefix="xmaps" %>
<html>
<head>
  <title>IntraNet</title>
</head>
<body>
  <xmaps:update scope="database" name="IntraNet" solution="example"/>
</body>
</html>
```

Event Handling

Any event defined in the diagram definition must make reference to a JavaScript function which will handle the event. This function will receive the event as parameter.

The example below shows a fragment of a diagram definition specifying that a select event shal be handled by the JavaScript function *onSelectedItem*.

```
<Diagram>
  …
  <On-Select>onSelectedItem</On-Select>
  …
</Diagram>
```

And the JavaScript function in the JSP that displays the diagram may look like:

```
<script>
function onSelectedItem(event)
{
  // get the event name
  var eventName = event.getName();
  // get the object in which the event has been fired
  var object = event.getSource();
  // get the object identifier
  var id = object.getId();
  …
}
</script>
```

As it can be seen, the event that is received as a parameter allows getting the name of the event that has been fired as well as a reference to the JavaScript object that has fired the event.

Saving Coordinates

When the diagram definition is obtained from the database the dock will contain an option for saving the diagram coordinates. By clicking on this option the diagram coordinates will be saved in the database and will be automatically applied to the elements in the diagram the next time it is displayed, skipping the coordinates from then diagram definition. If the next time new elements are found in the diagram, their coordinates from the diagram definition will be used.

Cross Launch

Service Activator provides a generic JSP to display diagrams and allows cross launching to it. This JSP is accessible through the URL

```
http://SAhost.mycompany.com:8080/activator/jsp/xmaps/xmaps.jsp
```

and accepts as request parameters all the attributes described in the section related to the tag library: *scope*, *id*, *name*, *solution*, *file*, *sort*, *gap*, *animate*, *store_in_session* and *session_key*.

As an example, this URL will cross launch to a JSP where the diagram with name *IntraNet* and solution name *example* will be dislayed:

```
http://localhost:8080/activator/jsp/login.jsp?hpsa_user=admin&hpsa_password
=admin&redirect=%2Factivator%2Fjsp%2Fxmaps%2Fxmaps.jsp%3Fscope=database%26n
ame=IntraNet%26solution=example%26animate=true
```

Cross launch is explained in *HP Service Activator, System Integrator’s Overview*.