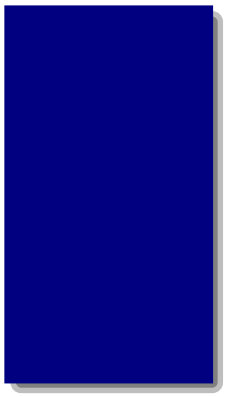


BroadVision®

InfoExchange Portal

InfoExchange Portal Developer's Guide

Version 6.0.0

BroadVision,® Inc.
585 Broadway
Redwood City, CA 94063
(650) 261-5100

Contents

1 Introduction

This manual provides Java, JavaScript and C++ developers with the information necessary to customize and extend BroadVision InfoExchange Portal. BroadVision InfoExchange Portal allows you to create views of your site that can be dynamically selected based on relevance to each site visitor.

BroadVision InfoExchange Portal includes the following:

- a Java component interface
- a JavaScript component interface
- a web-based administration tool
- the BVAdvenTech sample application

This chapter describes the concepts and features, as well as an overview on how to use the application programming interface (API) for BroadVision InfoExchange Portal.

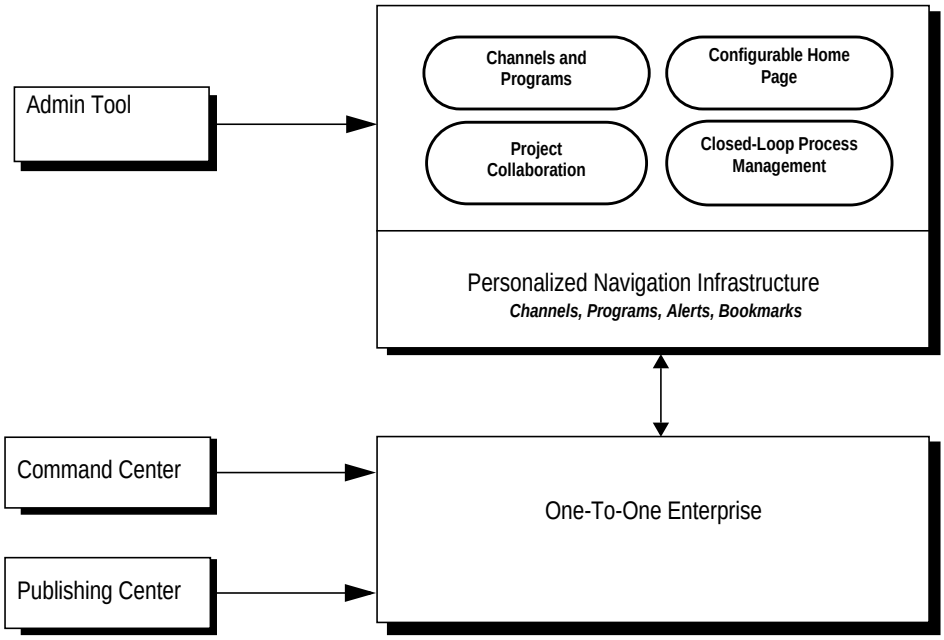
The following topics are discussed in this chapter:

BroadVision InfoExchange Portal overview

InfoExchange Portal helps you create views of your site that can be dynamically selected based on relevance to each site visitor. BroadVision InfoExchange Portal also provides functionality to enhance the relationship between a host enterprise, its customers, and its partners. Based on qualifiers, organizations, and other settings in the visitor's user account, using InfoExchange Portal you can:

- personalize the information delivered to each visitor
- create multiple, flexible channels to distribute the right information to the right recipients
- deliver information automatically to visitors entitled to receive it
- reduce the cost of supplying partners and customers with real-time information through distributed publishing with centralized controls
- automate business processes, such as managing and tracking leads
- improve employee productivity by providing easy access to relevant and timely content

The following diagram illustrates the main functional areas of InfoExchange Portal:



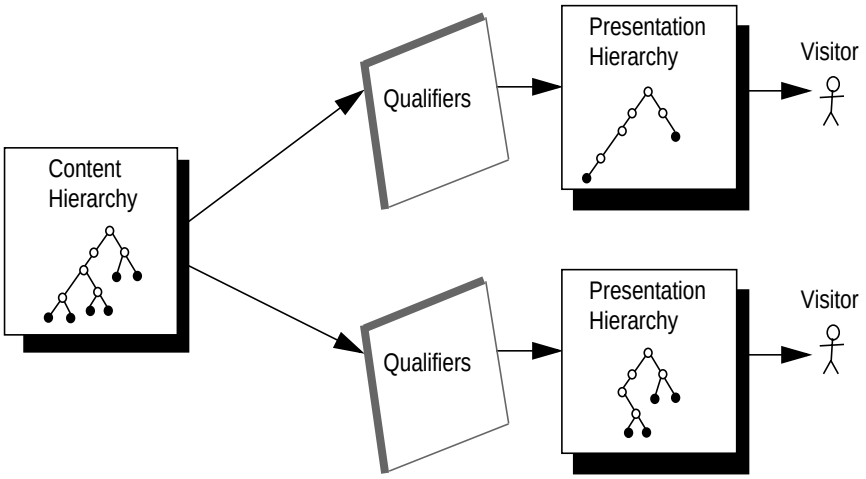
BroadVision InfoExchange Portal concepts

InfoExchange Portal contains the following configurable features:

- Personalized Navigation
- Organizations
- Configurable home pages
- Portlets
- Project collaboration pages
- Closed-loop process management
- Alerts, Bookmarks, and Search capabilities
- Site maintenance capabilities

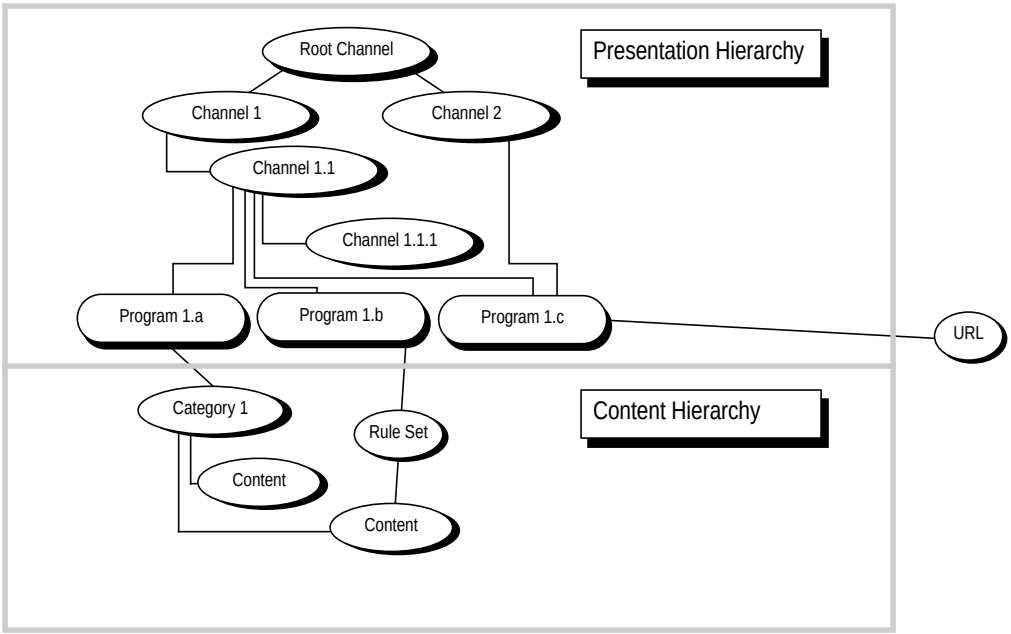
Personalized Navigation

Personalized navigation provides a way to customize information for a visitor. The view presented is independent of the organization of the content. The *presentation hierarchy* is defined by *channels* and *programs*. The *qualifiers* separate the presentation hierarchy from the content hierarchy.



Presentation Hierarchy

The *presentation hierarchy* describes the site views that can be presented to site visitors. It is comprised of *channels* and *programs* in a tree structure.



Channels

Channels define the presentation hierarchy structure to a visitor. Channels appear as links in a site's navigation menu. When the link is selected, the menu expands to reveal the sub channels and programs under the channel. The link also displays the channel's start page, which could display a description of the channel. When setting up a channel, the administrator specifies the channel's start page. In addition to showing presentation hierarchy organization, channels provide navigation links to programs.

Programs

Programs point to rules that retrieve content, external URLs, content, categories of content, or external information, such as a news feed. Programs appear as links in the site's navigation menu. When a program link is chosen, the link runs the program's start page, which typically shows links to content items. When you set up a program, you specify the program start page.

When you add a program, the Admin Tool prompts you for a *program type.* A program type specifies the type of content and rule set used for that program.

Qualifiers

Qualifiers are filters on the presentation hierarchy and the content hierarchy. Qualifiers are associated with each user template, visitor, channel, and program, and are used to further restrict a visitor's site view of channels and programs. The visitor's qualifier settings are dynamically matched against the qualifier settings for each channel, program, and content itemto determine the site view the visitor can see.

Each qualifier has one or more values that can be set. These qualifiers can either be set through the Admin Tool, or in some cases set by the visitor. For example, qualifiers can be "Region" with values such as "Europe" or "Asia," or "Language" with values such as "English" and "French."

Qualifiers are defined for the site by the Site Administrator using the Admin Tool. Qualifiers are set for channels and programs by the Site Administrator or Channel Administrator; visitor-specific qualifiers are set for visitors by the Site Administrator or Service Administrator using the Admin Tool. The Site Administrator or User Administrator can also define which qualifiers can be set by the visitor.

By matching qualifiers, InfoExchange Portal determines which channels and programs a visitor can see and therefore access.

Content Hierarchy

Content in a One-To-One Enterprise system is organized in a hierarchy of categories and content. The Command Center and Publishing Center manage the categories and content.

Categories

Categories define the hierarchy in which content is organized for publishing and targeting.

Content

Content is the information in your site. Content can live in more than one category.

Organizations

Organizations let you define a hierarchy of divisions, departments, groups, and so forth of user accounts. Organizations are defined for the site an administrator using the Admin Tool. While qualifiers must be administered by a Site Administrator or Service Administrator, a visitor who is not a Site Administrator or Service Administrator can administer an organization.

Configurable home pages

Visitor configurable home pages enable visitors to customize the information they see on their home page. There are several levels at which pages can be customized and configured:

- site administrator configuration
- visitor configuration

Sites created with InfoExchange Portal can have different types of home pages. For example, the home page for a partner might be a different type of home page than the home page for a customer.

Visitors log into their user accounts to access information on their home pages. A *user account* for each visitor is created by an InfoExchange Portal Site Administrator or User Administrator and assigned to a *user template*. The user template defines the page type and home page for each visitor.

The user template sets default qualifiers, bookmarked content items, and access rights for each new visitor assigned a user account.

Home Page Concepts

The *home page* is a script that personalizes a visitor's view of the site. This home page can contain the visitor's alerts, as well as default programs and channels as filtered through qualifiers. The visitor's home page is assigned using the Admin Tool.

There are three major configurable parts in a home page:

- **Page type**. A *page type* determines which script is used to generate the home page for a visitor.
- **Block**. A *block* is a logical group of topics that can be placed on a home page. There are three types of blocks:
  - **Required** - seen by everyone
  - **Default** - seen by new users
  - **Optional** - configurable by the visitor

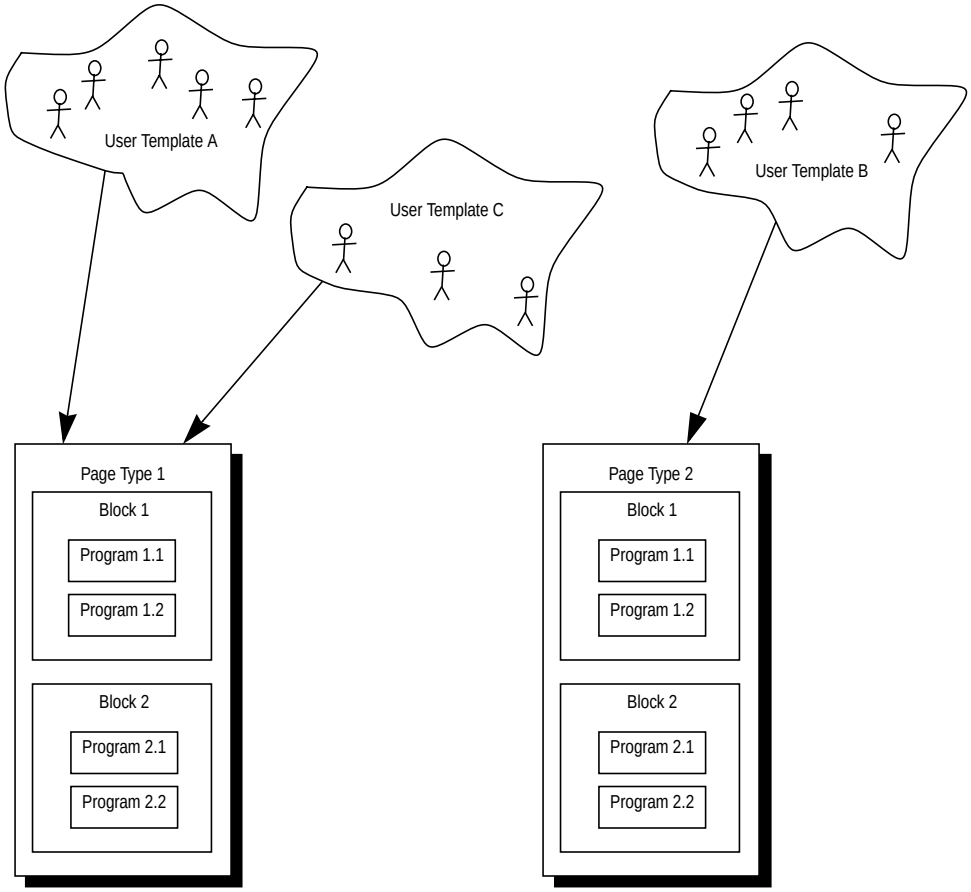
  Visitors can customize the default and optional blocks.
- **Topic**. A *topic* is a program that can display content or other information on a page, such as a news feed.

Blocks and topics are set up in the Admin Tool.

Home pages also support *transient guests*, or visitors to a site who have not logged in. Each visitor can be associated with only one user template.

User templates

When you create a user account, the visitor inherits the qualifiers and access rights from the user template assigned to that account. The Site Administrator can override the settings inherited from the user template for any user account.



You define user templates with the Admin Tool (see the *InfoExchange Administrator's Guide* for more information). The sample data shipped with the product includes several predefined user templates, such as Executive, Customer, and Techie.

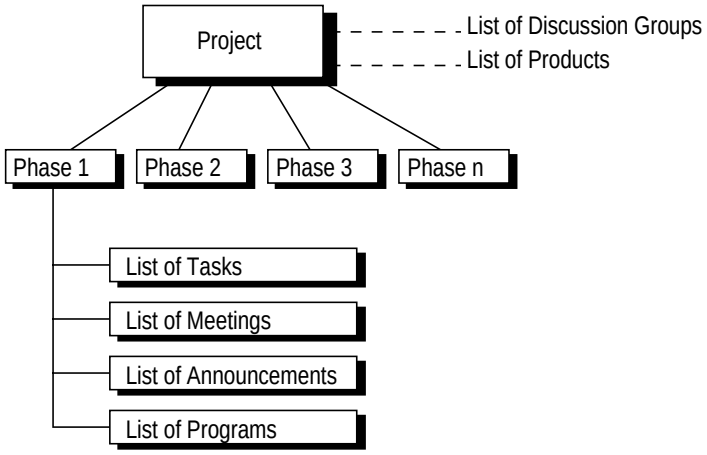
Portlets

A *portlet* is a reusable interface component that provides access to web-based resources. You can use portlets to customize and extend BroadVision InfoExchange Portal to display and integrate external or legacy data sources that reside outside of the One-To-One Enterprise database. Examples of these sources are a Reuters news feed, stock quotes, and MS Exchange Server. Site administrators can manage the access and display of a portlet. Portlets can be set up so visitors can set parameters within them, such as an MS Outlook user name and password.

Project collaboration pages

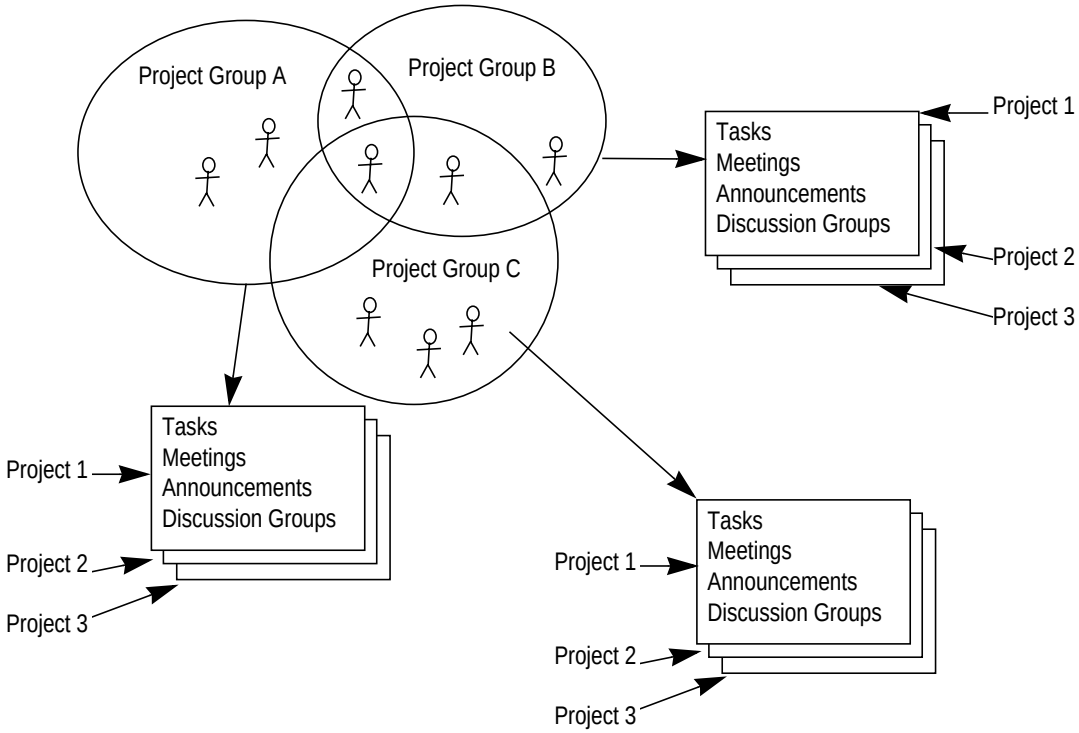
The *project collaboration page* is a web page similar to a bulletin board. It is created by an employee, such as a sales representative or engagement manager, for sharing information about a project with fellow employees, customers, and partners. A collaboration page pertains to a single project with a life cycle that contains one or more *phases*. A phase is a stage of the project, such as design, beta, or testing.

Participants can create tasks, meetings, announcements, and participate in threaded discussion groups.



More than one phase can be in progress at one time. Information that can be shared about a project includes tasks, meetings, and announcements. *Tasks* are essentially a To-Do list.

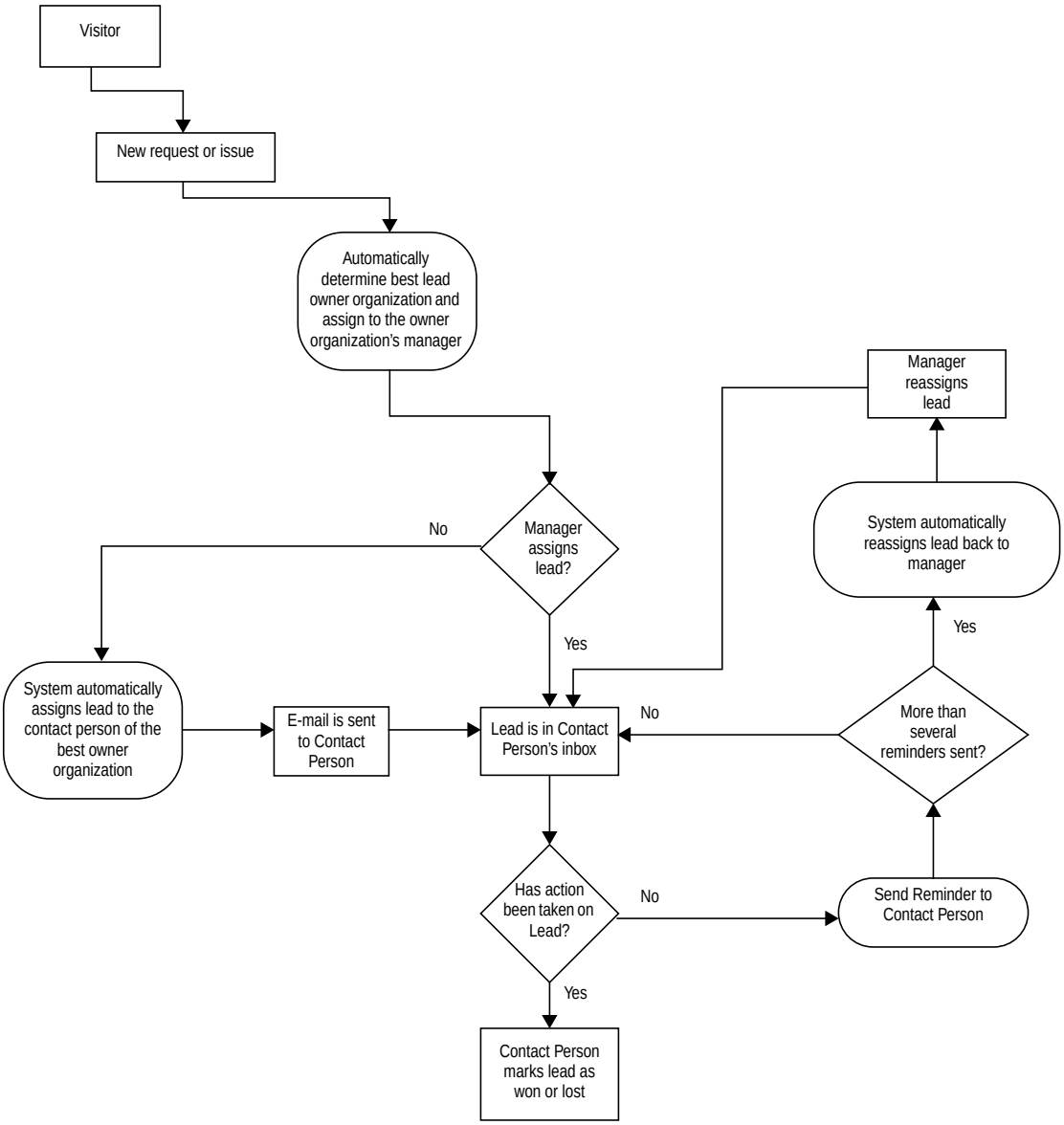
Tasks, meetings, announcements and discussion groups can belong to only one project. However, the people sharing a project can belong to more than one project.



See Chapter 5, “Customizing the Project Collaboration Page” for more information.

Closed-loop process management

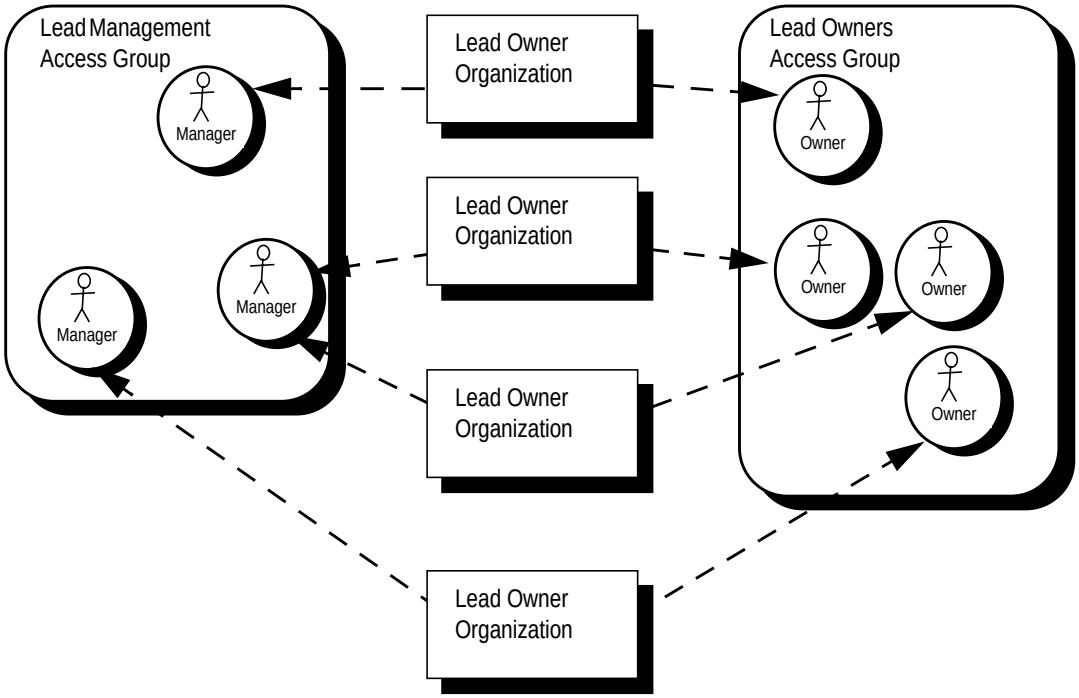
Closed-loop process management provides a method for tracking and following up on specific types of information. This information can be a sales lead, trouble ticket or some other implementation of the function. Items advancing through the closed loop management process are never lost, as the system tracks the progress and individuals involved.



Visitors enter a request or an issue through the web. The request or issue is automatically routed to an owner based on workflow and rules. The workflow is defined using the Publishing Center; the rules are created using the Command Center. The owner is then notified and reminded of new action items. If the owner is unable to take action on the action items, the manager is notified after a certain period of time has elapsed, again based on workflow and rules, and the manager can reassign the action items.

For example, sales leads can be created after a visitor accesses a site. When a visitor requests information from a site, they are taken to a form, and asked to provide some background information.

Submitting the form creates a lead content item which is initially routed to a lead manager within the company running the site and then to a lead owner. If the lead owner is unable to immediately follow up on the lead, the lead can be quickly reassigned.



See Chapter 6, “Customizing Closed-loop Process Management” for more information.

Alerts, Bookmarks, and Search capabilities

Alerts are messages generated when a particular content item changes. The metadata on the database (specifically, the `LAST_MOD_TIME` column) describing the content triggers the Alert when changed. If the content item is external (a file, for example), changes to its contents do not trigger an Alert until the metadata on the database is updated.

Bookmarks are user configurable links to specific channels, program or content.

Search allows the visitor to find specific content using Verity’s full text search.

Site maintenance capabilities

InfoExchange Portal has several features that enable you to monitor and improve performance of your InfoExchange Portal site.

The *filter cache* lets you set the size of the cache used for storing the qualifiers of content. By using a filter cache you can speed up performance of your InfoExchange Portal site by not having to look up the qualifiers on content items multiple times.

The *home page cache* lets you enable, disable, and set the size of the cache used for recurring items displayed on a visitor's home page. Like the filter cache, enabling the home page cache speeds up performance of an InfoExchange Portal site.

InfoExchange Portal also has a series of reports included that return site information and statistics. Using this information, a site administrator can adjust parameters to improve an InfoExchange Portal site.

The BVAdvenTech sample application

InfoExchange Portal is shipped with a sample application you can modify called BVAdvenTech. The application comes with data for the following site visitors:

Name	Password	Description	Administrative Functions	User Template
miles	miles	sales manager	Site Admin, Publishing Center User, Project Admin, Lead Owner Admin	Manager
cesar	cesar	visitor	none	Customer
suzannah	suzannah	sales manager	Channel Admin, Publishing Center User, Project Admin, Lead Owner Admin	Manager
robyn	robyn	sales agent	none	Partner
martha	martha	sales manager	Publishing Center User, Project Admin	Business
sara	sara	sales manager	Publishing Center User, Project Admin, Lead Owner Admin	Business
sid	sid	non-sales employee	none	Techie
teri	teri	non-sales employee	Project Admin	Techie
lin	lin	contact at a sales agent reporting to miles	none	Partner
don	don	contact at a sales agent reporting to miles	none	Partner
mat	mat	contact at a sales agent reporting to miles	none	Partner
joe	joe	contact at a sales agent reporting to sara	none	Partner
lisa	lisa	contact at a sales agent reporting to sara	none	Partner
evan	evan	contact at a sales agent reporting to sara	none	Partner
phil	phil	contact at a sales agent reporting to suzannah	none	Partner
dave	dave	contact at a sales agent reporting to suzannah	none	Partner

You can log into the site created with the BVAdvenTech sample application to see how the site appears differently to each of these visitors.

Throughout this document examples are taken from the BVAdvenTech sample application. You can create your own site by either modifying the BVAdvenTech sample application or by creating a site from scratch.

Related documentation

There are three other documents for InfoExchange Portal:

- *InfoExchange Portal Installation and Release Notes*
- *InfoExchange Portal Concepts Guide*
- *InfoExchange Administrator's Guide*

This guide, the *InfoExchange Portal Developer's Guide*, provides an extensive discussion of how to use the BroadVision-supplied C++ components in JavaScript scripts or Java server pages to customize or extend an InfoExchange Portal site.

2 Navigation and filtering methods

BroadVision One-To-One Enterprise and the applications that run on top of it use JavaScript and Java interfaces to access C++ functionality provided by the One-To-One application and components. This chapter is a reference to the methods and attributes provided by the component interfaces for channels, programs, and navigation filters of InfoExchange Portal.

These are the component interfaces for channels, programs, and navigation filters:

When you write your own scripts, besides testing for general error conditions, make sure you test for the following error conditions specific to InfoExchange Portal:

- content, channel, or program that is off-line
- content, channel, or program that is deleted
- qualifier mismatch, causing a previously visible channel or program to now be invisible to a visitor

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; Java classes reside in the `com.broadvision.ieportal.components` Java package.

BVI_KMProgramManager

When a method in this interface returns a BVI_ValueList, the BVI_Value objects it refers to contain objects such as BVI_KMProgram or BVI_KMChannel. To extract these objects for use in a script, loop through the list using the BVI_ValueList.get method and the BVI_Value.objectValue attribute. See the *Developer's Guide to Components and Scripts* for information about the BVI_ValueList and BVI_Value objects.

The BVI_Value object can return an object other than a BVI_KMProgram object or BVI_KMChannel object when methods return a BVI_ValueList object. For example, the BVI_KMProgramManager.getBookmarkedContentList() method returns a BVI_ValueList object; each BVI_Value object refers to a BVI_Properties object.

Bookmark methods that have the word "default" in their name return the channels, programs, and content items the administrator has assigned to the visitor through the user template. All other bookmark methods return the channels, programs, and content items the visitor has selected.

The user template contains the relative paths for the profile page and the matching page. Note that a visitor has only one assigned user template. (See "User profile and related tables" on page 291 and the description of BV_KM_USER_TYPE for related information.)

This interface does not support dynamic properties.

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; Java classes reside in the `com.broadvision.ieportal.components` Java package.

Method	Functionality
creator()	Creates a new BVI_KMProgramManager object.
creator()	Creates a copy of the specified BVI_KMProgramManager object.
clone()	Creates a copy of this BVI_KMProgramManager object (JavaScript).
clone_J()	Creates a copy of this BVI_KMProgramManager object (Java).
getHomePagePath()	Retrieves the path to the home page for the specified visitor and service. **Obsolete**; use BVI_EPHomePage::getUserPageTypeSettings().
getHomeChannelList()	Retrieves the top-level channels visible to the specified visitor in the specified service.
getProfilePagePath()	Retrieves the path to the personal profile page for the specified visitor and service.
getMatchingPagePath()	Retrieves the path to the matching profile page for the specified visitor and service.
getChannel()	Retrieves information about the specified channel.
getChannelList()	Retrieves the sub channels for the specified channel where channelId is the channel ID or full channel path name. When '`ep_enable_qualifiers`' is set to "1", the returned channel list has already been filtered based on the qualifier setting on channels and visitor profile.
getProgram()	Retrieves information about the specified program if the program is on-line.
getProgramById()	Retrieves information about the specified program whether the program is on-line or off-line.

Method	Functionality
getProgramList()	Retrieves the programs associated with the specified channel. When '`ep_enable_qualifiers`' is set to "1", the returned program list has already been filtered based on the qualifier setting on channels and visitor profile.
getProgramListByName()	Finds programs with names that contain a specified substring. When '`ep_enable_qualifiers`' is set to "1", the returned program list has already been filtered based on the qualifier setting on channels and visitor profile.
getContentList()	Retrieves the content items referenced by the specified program.
isContentReadable()	Checks whether a content item is readable by this visitor.
filterContentList()	Makes a copy of the specified content list that contains only those items for which the specified visitor has read privileges.
getChannelPathList()	Retrieves the list of channels in the path to the specified channel.
getUserStatus()	Retrieves the visitor's status.
isAlertContentType()	Checks whether an alert can be set for a content type.
getSubtypePagePath()	Retrieves the path to the page display script for the specified content subtype.
findContent()	Finds content with names that contain a specified substring.
bookmarkChannel()	Sets or removes a channel bookmark for the specified visitor.
getBookmarkedChannelList()	Retrieves the list of bookmarked channels for the specified visitor.
getDefaultBookmarkedChannelList()	Retrieves the list of default bookmarked channels for the specified visitor.
isBookmarkedChannel()	Checks whether a channel is bookmarked for the specified visitor.
bookmarkProgram()	Sets or removes a program bookmark for the specified visitor.
getBookmarkedProgramList()	Retrieves the list of bookmarked programs for the specified visitor (`userId`).
getDefaultBookmarkedProgramList()	Retrieves the list of default bookmarked programs for the specified visitor.
isBookmarkedProgram()	Checks whether a program is bookmarked for the specified visitor.
bookmarkContent()	Sets or removes a content bookmark for the specified visitor.
getBookmarkedContentList()	Retrieves the list of bookmarked content for the specified visitor.
getDefaultBookmarkedContentList()	Retrieves the list of default bookmarked content for the specified visitor.
isBookmarkedContent()	Checks whether a content item is bookmarked for the specified visitor.
isChannelVisible()	Checks whether a channel is visible to the specified visitor.
isProgramVisible()	Checks whether a program is visible to the specified visitor.

BVI_KMProgramManager::creator()

Creates a new BVI_KMProgramManager object.

JavaScript
```
BVI_KMProgramManager creator();
```

Java
```
public BVI_KMProgramManager creator()
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError.

Parameters None.

Return value A reference to a new BVI_KMProgramManager object; null if an error occurs.

BVI_KMProgramManager::creator()

Creates a copy of the specified BVI_KMProgramManager object.

JavaScript
```
BVI_KMProgramManager creator(BVI_KMProgramManager ref);
```

Java
```
public BVI_KMProgramManager creator(BVI_KMProgramManager ref)
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError.

Parameters

`ref`	A BVI_KMProgramManager object.

Return value Returns a BVI_KMProgramManager object; in JavaScript, returns null if an error occurs.

BVI_KMProgramManager::clone()

Creates a copy of this BVI_KMProgramManager object.

JavaScript
```
BVI_KMProgramManager clone();
```

Parameters None.

Return value Returns a BVI_KMProgramManager object; returns null if an error occurs.

BVI_KMProgramManager::clone_J()

Creates a copy of this BVI_KMProgramManager object.

Java

```
public final BVI_KMProgramManager clone_J()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value Returns a BVI_KMProgramManager object.

BVI_KMProgramManager::getHomePagePath()

Retrieves the path to the home page for the specified visitor (`userId`) and service. **Obsolete**; use BVI_EPHomePage::getUserPageTypeSettings().

BVI_KMProgramManager::getHomeChannelList()

Retrieves the top-level channels visible to the specified visitor (`userId`) in the specified service.

JavaScript

```
BVI_ValueList getHomeChannelList(
    string serviceId,
    long userId );
```

Java

```
public final BVI_ValueList getHomeChannelList(
    String serviceId,
    long userId)
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	ID for the current service.
`userId`	ID for the current visitor.

Return value A list of BVI_Value objects, each containing one BVI_KMChannel object.

Remarks To extract the BVI_KMChannel objects for use in a script, loop through the list and use the BVI_Value.objectValue attribute. The home channels are the top-level channels visible to the specified visitor in the specified service.

See also `BVI_KMProgramManager::getChannelList()` on page 26.

BVI_KMProgramManager::getProfilePagePath()

Retrieves the path to the personal profile page for the specified visitor (`userId`) and service.

JavaScript

```
string getProfilePagePath(
    string serviceId,
    long userId );
```

Java

```
public final String getProfilePagePath(String serviceId, long userId)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID for the current service.
`userId`	An integer; the ID for the current visitor.

Return value

A string containing the path to a personal profile page script.

Example

The following example is from `/adventech/scripts/common/changeProfile.jsp` and works with getMatchingPagePath(). `km_progMgr` is an instance of the BVI_KMProgramManager component.

```
<%
//
// Get the user's profile page (based on the user's assigned
// User Template).
//
var profilePage = km_progMgr.getProfilePagePath(serviceId, userId);
if (isEmptyString(profilePage)) {
    // assign default profile destination
    profilePage = '/adventech/scripts/common/changeProfile.jsp';
}
%>
<a href="<%= fastconcat(Request.SCRIPT_NAME, profilePage, "?",
linkString) %>"
    class="programLinks">Change personal details</a>
```

Remarks

The returned path is retrieved from the visitor's user template defined in the BV_KM_USER_TYPE table. (See "User profile and related tables" on page 291.) The returned path is relative to the script root and specifies the actual page script as in the following:

```
'/adventech/scripts/common/changeProfile.jsp'
```

BVI_KMProgramManager::getMatchingPagePath()

Retrieves the path to the matching profile page for the specified visitor (`userId`) and service.

JavaScript

```
string getMatchingPagePath( string serviceId, long userId );
```

Java

```
public final String getMatchingPagePath(String serviceId, long userId)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID for the current service.
`userId`	An integer; the ID for the current visitor.

Return value

A string containing the path to a matching profile page script.

Example

The following example is from `/adventech/scripts/common/changeProfile.jsp` and works with getProfilePagePath(). `km_progMgr` is an instance of the BVI_KMProgramManager component.

```
//
// Get the user's matching profile page (based on the user's
// assigned User Template).
//

var matchingPage = km_progMgr.getMatchingPagePath(serviceId, userId);

if (isEmptyString(matchingPage)) {
    // assign default matching destination
    matchingPage = '/adventech/scripts/common/changeMatch.jsp';
}
<a href="<%= fastconcat(Request.SCRIPT_NAME, matchingPage, "?",
linkString) %>"
    class="programLinks">Change matching information</a>
```

Remarks

The returned path is retrieved from the visitor's user template defined in the BV_KM_USER_TYPE table. (See "User profile and related tables" on page 291.) The returned path is relative to the script root and specifies the actual page script as in:

```
'/adventech/scripts/common/changeMatch.jsp'
```

BVI_KMProgramManager::getChannel()

Retrieves information about the specified channel.

JavaScript
```
BVI_KMChannel getChannel( string serviceId, string channelId );
```

Java
```
public final BVI_KMChannel getChannel(String serviceId, String channelId)
```

Java exception — Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID for the current service.
`channelId`	A string; the full path name for a channel.,

Return value — A BVI_KMChannel object.

BVI_KMProgramManager::getChannelList()

Retrieves the subchannels for the specified channel where `channelId` is the channel ID or full channel path name. When `'ep_enable_qualifiers'` is set to 1, the returned channel list has already been filtered based on the qualifier setting on channels and visitor profile.

JavaScript
```
BVI_ValueList getChannelList(
    string serviceId,
    long userId,
    string channelId );
```

Java
```
public final BVI_ValueList getChannelList(
    String serviceId,
    long userId,
    String channelId );
```

Java exception — Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.
`channelId`	A string; ID or full path name for a channel..

Return value — A list of `BVI_Value` objects, each containing one `BVI_KMChannel` object.

Remarks — To extract the `BVI_KMChannel` objects for use in a script, loop through the list and use the `BVI_Value.objectValue` attribute.

BVI_KMProgramManager::getProgram()

Retrieves information about the specified program if the program is on-line.

JavaScript
```
BVI_KMProgram getProgram( string serviceId, string programId );
```

Java
```
public final BVI_KMProgram getProgram(String serviceId, String programId)
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`programId`	A string; ID or name for a program.

Return value A BVI_KMProgram object.

BVI_KMProgramManager::getProgramById()

Retrieves information about the specified program whether the program is on-line or off-line.

JavaScript
```
BVI_KMProgram getProgram(string serviceId, string programId);
```

Java
```
public final BVI_KMProgram getProgram(String serviceId, String programId)
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`programId`	A string; ID or name for a program.

Return value A BVI_KMProgram object.

BVI_KMProgramManager::getProgramList()

Retrieves the programs associated with the specified channel. When `'ep_enable_qualifiers'` is set to 1, the returned program list has already been filtered based on the qualifier setting on channels and visitor profile.

JavaScript
```
BVI_ValueList getProgramList(
    string serviceId,
    long userId,
    string channelId);
```

Java
```
public final BVI_ValueList getProgramList(
    String serviceId,
    long userId,
    String channelId);
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.
`channelId`	A string; ID or full path name for a channel..

Return value A list of `BVI_Value` objects, each containing one BVI_KMProgram object.

Remarks To extract the BVI_KMProgram objects for use in a script, loop through the list and use the `BVI_Value.objectValue` attribute.

BVI_KMProgramManager::getProgramListByName()

Finds programs with names that contain a specified substring. When `'ep_enable_qualifiers'` is set to 1, the returned program list has already been filtered based on the qualifier setting on channels and visitor profile.

JavaScript
```
BVI_ValueList getProgramListByName(
    string service,
    long userId,
    string nameLikeString,
    long maxCount,
    long maxCountBeforeFiltering);
```

Java
```
public final BVI_ValueList getProgramListByName(
    String serviceId,
    long userId,
    String nameLikeString,
    long maxCount,
    long maxCountBeforeFiltering)
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.
`nameLikeString`	A string; the substring used for search through program names.
`maxCount`	An integer; the maximum count of programs to return
`maxCountBeforeFiltering`	An integer; the maximum count of programs to retrieve from the database.

Return value A list of BVI_Value objects, each containing one BVI_KMProgram object.

Remarks

This method is case insensitive. This method retrieves up to `maxCountBeforeFiltering` number of programs containing the substring `nameLikeString`, and then applies the navigation filters and returns the first `maxCount` programs in the filtered list. To retrieve all, set `maxCount` and `maxCountBeforeFiltering` to 0, but performance may be slow if there is a large number of programs in the database.

BVI_KMProgramManager::getContentList()

Retrieves the content items referenced by the specified program.

JavaScript

```
BVI_ContentList getContentList(
    string serviceId,
    long userId,
    string programId,
    BVI_SessnProf sprof,
    long max_count );
```

Java

```
public final BVI_ContentList getContentList(
    String serviceId,
    long userId,
    String programId,
    BVI_SessnProf sprof,
    long max_count)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.
`programId`	A string; ID or name for a program.
`sprof`	Session profile for this visitor.
`maxCount`	An integer; the maximum count of programs to return

Return value

A list of `BVI_Content` objects.

Remarks

When, in the `bv1to1.conf` file, `ep_enable_qualifiers` is set to “1” and the current content type is enabled for filtering (in `ep_filter_enabled_cnt_types`), the returned content list is filtered based on the qualifier settings in the content and visitor profile.

BVI_KMProgramManager::isContentReadable()

Checks whether a content item is readable by this visitor (`userId`).

JavaScript

```
boolean isContentReadable(
    long userId,
    string serviceId,
    string contentType,
    long contentOID );
```

Java

```
public final boolean isContentReadable(
    long userId,
    String serviceId,
    String contentType,
    long contentOID)
```

Java exception — Throws BVWComponentException.

Parameters

`userId`	An integer; user ID for the current visitor.
`serviceId`	A string; the ID or name of the current service.
`contentType`	A string; the type of the content item, for example, EDITORIAL.
`contentOID`	An integer; ID for the content item.

Return value — True if the visitor has read privileges for the specified content item.

Remarks — This method uses the functionality underlying the Publishing Center to determine if the visitor has read privileges.

This method does *not* use the BroadVision One-To-One Enterprise `BVI_ACLManager` functionality.

BVI_KMProgramManager::filterContentList()

Makes a copy of the specified content list that contains only those items for which the specified visitor (`userId`) has read privileges.

JavaScript

```
BVI_ContentList filterContentList(
    long userId,
    BVI_ContentList contentList );
```

Java

```
public final BVI_ContentList filterContentList(
    long userId,
    BVI_ContentList contentList)
```

Java exception — Throws BVWComponentException.

Parameters

`userId`	An integer; user ID for the current visitor.
`contentList`	The list to be filtered.

Return value — A BVI_ContentList containing only those items from `contentList` to which the visitor has read access.

BVI_KMProgramManager::getChannelPathList()

Retrieves the list of channels in the path to the specified channel.

JavaScript

```
BVI_ValueList getChannelPathList(
    string service,
    string channelId);
```

Java

```
public final BVI_ValueList getChannelPathList(
    String service,
    String channelId)
```

Java exception

Throws BVWComponentException.

Parameters

`service`	A string; the name of the current service.
`channelId`	A string; the ID or full path name for the channel.

Return value

A list of BVI_Value objects, each of which contains a BVI_KMChannel object.

Remarks

Each returned BVI_KMChannel object contains the full path for each parent channel along the path back to the root channel. You can use this method in building a channel navigation bar. For example, if the channel path looks something like this:

```
root->computers->accessories->modems
```

This method returns two BVI_KMChannel objects with respective attributes that can be used to build links to the pages that represent the channels between the current channel and the root channel. That is, one object contains the information for the computers channel and one object contains the information for the accessories channel.

To extract the BVI_KMChannel objects for use in a script, loop through the list and use the BVI_Value.objectValue attribute. Then use the BVI_KMChannel::channelId and BVI_KMChannel::channelName attributes to create links.


The returned list does not include the root channel.

BVI_KMProgramManager::getUserStatus()

Retrieves the visitor's status.

JavaScript

```
long getUserStatus(string service, long userId);
```

Java

```
public final long getUserStatus(String service, long userId)
```

Java exception

Throws BVWComponentException.

Parameters

`service`	A string; the name of the current service.
`userId`	An integer; the user ID for the current visitor.

Return value — A long integer representing the status.

Remarks — The returned status can have the following values:

Value	Meaning
0	valid active user account
1	visitor has an invalid user template
2	non-active user ID; visitor is not allowed to login to an InfoExchange Portal site

BVI_KMProgramManager::isAlertContentType()

Checks whether an alert can be set for a content type.

JavaScript

```
boolean isAlertContentType(string contentType);
```

Java

```
public final boolean isAlertContentType(String contentType)
```

Java exception — Throws BVWComponentException.

Parameters

`contentType`	A string; the type of the content item, for example, EDITORIAL.

Return value — True, if `contentType` is one of the content types defined for `km_alert_content_type` in `$BV1TO1_VAR/etc/bv1to1.conf`.

Remarks — The `km_alert_content_type` is set in `$BV1TO1_VAR/etc/bv1to1.conf` and specifies which content types the InfoExchange Portal alert plug-in processes.

BVI_KMProgramManager::getSubtypePagePath()

Retrieves the path to the page display script for the specified content subtype.

The page is the content display script for content in that subtype. Subtypes are a feature of the Publishing Center. See the *Publishing Center Developer's Guide* for more information on subtypes.

JavaScript

```
string getSubtypePagePath(
    string serviceId,
    string contentType,
    string subtypeId );
```

Java
```
public final String getSubtypePagePath(
    String serviceId,
    String contentType,
    String subtypeId)
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string; the ID or name of the current service.
contentType	A string; the type of the content item, for example, EDITORIAL.
subtypeId	A string; the ID or name of the subtype.

Return value A string, containing a path to a page script.

Remarks The returned path is relative to the script root.

BVI_KMProgramManager::findContent()

Finds content with names that contain a specified substring.

JavaScript
```
BVI_ContentList findContent(
    string service,
    string contentType,
    string nameLikeString,
    long maxItems);
```

Java
```
public final BVI_ContentList findContent(
    String service,
    String contentType,
    String nameLikeString,
    long maxItems)
```

Java exception Throws BVWComponentException.

Parameters

service	A string; the ID or name of the current service.
contentType	A string; the name or ID of the content type.
nameLikeString	A string; the substring to search through content names for.
maxItems	An integer; the maximum number of content items to return.

Return value A list of BVI_Content objects.

Remarks This method is case insensitive.

BVI_KMProgramManager::bookmarkChannel()

Sets or removes a channel bookmark for the specified visitor (user ID).

JavaScript

```
void bookmarkChannel(
    string serviceId,
    long userId,
    string channelId,
    boolean flag );
```

Java

```
public final void bookmarkChannel(
    String serviceId,
    long userId,
    String channelId,
    boolean flag)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.
`channelId`	A string; ID or full path name for a channel..
`flag`	A boolean; set this to true to place a channel bookmark, false to remove a channel bookmark.

Return value

None.

BVI_KMProgramManager::getBookmarkedChannelList()

Retrieves the list of bookmarked channels for the specified visitor.

JavaScript

```
BVI_ValueList getBookmarkedChannelList(
    string serviceId,
    long userId );
```

Java

```
public final BVI_ValueList getBookmarkedChannelList(
    String serviceId,
    long userId)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.

Return value

A list of BVI_Value objects, each containing a BVI_KMChannel object.

Remarks

To extract the BVI_KMChannel objects for use in a script, loop through the list using the BVI_ValueList.get method and the BVI_Value.objectValue attribute. Then use the BVI_KMChannel::channelId and BVI_KMChannel::channelName attributes to create links.

BVI_KMProgramManager::getDefaultBookmarkedChannelList()

Retrieves the list of default bookmarked channels for the specified visitor.

JavaScript

```
BVI_ValueList getDefaultBookmarkedChannelList(
    string serviceId,
    long userId );
```

Java

```
public final BVI_ValueList getDefaultBookmarkedChannelList(
    String serviceId,
    long userId)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.

Return value

A list of BVI_Value objects, each containing a BVI_KMChannel object.

Remarks

To extract the BVI_KMChannel objects for use in a script, loop through the list using the BVI_ValueList.get method and the BVI_Value.objectValue attribute. Then use the BVI_KMChannel::channelId and BVI_KMChannel::channelName attributes to create links.

The retrieved list of default bookmarked channels is based on the visitor's user template.

BVI_KMProgramManager::isBookmarkedChannel()

Checks whether a channel is bookmarked for the specified visitor (`userId`).

JavaScript

```
boolean isBookmarkedChannel(
    string serviceId,
    long userId,
    string channelId );
```

Java

```
public final boolean isBookmarkedChannel(
    String serviceId,
    long userId,
    String channelId)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.
`channelId`	A string; the ID or full path name for the channel.

Return value — True if the channel is bookmarked for the specified visitor; otherwise false.

BVI_KMProgramManager::bookmarkProgram()

Sets or removes a program bookmark for the specified visitor.

JavaScript

```
void bookmarkProgram(
    string serviceId,
    long userId,
    string programId,
    boolean flag);
```

Java

```
public final void bookmarkProgram(
    String serviceId,
    long userId,
    String programId,
    boolean flag)
```

Java exception — Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.
`programId`	A string; the ID or full path name for the program.
`flag`	A boolean; set this to true to place a program bookmark, false to remove a program bookmark.

Return value — None.

BVI_KMProgramManager::getBookmarkedProgramList()

Retrieves the list of bookmarked programs for the specified visitor (`userId`).

JavaScript

```
BVI_ValueList getBookmarkedProgramList(
    string serviceId,
    long userId);
```

Java

```
public final BVI_ValueList getBookmarkedProgramList(
    String serviceId,
    long userId)
```

Java exception — Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.

Return value — A list of BVI_Value objects, each containing a BVI_KMProgram object.

Remarks — To extract the BVI_KMProgram objects for use in a script, loop through the list using the BVI_ValueList.get method and the BVI_Value.objectValue attribute. Then use the BVI_KMProgram::programId and BVI_KMProgram::programName attributes to create links.

Example — This code segment demonstrates the essentials of the loop.

```
for (i = 0; i < programLen; i++)
{
item = programList.get(i);
program = item.objectValue;
...
... // do something to display the program links
}
```

BVI_KMProgramManager::getDefaultBookmarkedProgramList()

Retrieves the list of default bookmarked programs for the specified visitor.

JavaScript

```
BVI_ValueList getDefaultBookmarkedProgramList(
    string serviceId,
    long userId);
```

Java

```
public final BVI_ValueList getDefaultBookmarkedProgramList(
    String serviceId,
    long userId)
```

Java exception — Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.

Return value — A list of BVI_Value objects, each containing a BVI_KMProgram object.

Remarks — To extract the BVI_KMProgram objects for use in a script, loop through the list using the BVI_ValueList.get method and the BVI_Value.objectValue attribute. Then use the BVI_KMProgram::programId and BVI_KMProgram::programName attributes to create links.

The retrieved list of default bookmarked programs is based on the visitor's user template. (See "User profile and related tables" on page 291.)

See also BVI_KMProgramManager::getBookmarkedProgramList()

BVI_KMProgramManager::isBookmarkedProgram()

Checks whether a program is bookmarked for the specified visitor.

JavaScript

```
boolean isBookmarkedProgram(
    string serviceId,
    long userId,
    string programId);
```

Java

```
public final boolean isBookmarkedProgram(
    String serviceId,
    long userId,
    String programId)
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string; the ID or name of the current service.
userId	An integer; user ID for the current visitor.
programId	A string; the ID or name for the program.

Return value True if the program is bookmarked for the specified visitor; false otherwise.

BVI_KMProgramManager::bookmarkContent()

Sets or removes a content bookmark for the specified visitor.

JavaScript

```
void bookmarkContent(
    string serviceId,
    long userId,
    string contentType,
    long contentOID,
    boolean flag );
```

Java

```
public final void bookmarkContent(
    String serviceId,
    long userId,
    String contentType,
    long contentOID,
    boolean flag)
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.
`contentType`	A string; the ID or name for the content type.
`contentOID`	An integer; the content OID.
`flag`	A boolean; set this to true to place a contact bookmark, false to remove a contact bookmark.

Return value None.

BVI_KMProgramManager::getBookmarkedContentList()

Retrieves the list of bookmarked content for the specified visitor.

JavaScript

```
BVI_ValueList getBookmarkedContentList(
    string serviceId,
    long userId);
```

Java

```
public final BVI_ValueList getBookmarkedContentList(
    String serviceId,
    long userId)
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.

Return value A list of BVI_Value objects, each containing a BVI_Properties object.

Remarks To extract the BVI_Properties objects for use in a script, loop through the list using the BVI_ValueList.get method and the BVI_Value.objectValue attribute. The BVI_Properties objects contain three long values: OID, CONTENT_TYPE, and SERVICE_ID. Pass these three values to BVI_ContentManager.contentByOID() to retrieve the actual content.

Example This code segment demonstrates the essentials of the loop.

```
for (i = 0; i < contentLen; i++)
{
item = contentList.get(i);
uid = item.objectValue;
cnt = contentMgr.contentByOID(uid.OID, uid.SERVICE_ID, uid.CONTENT_TYPE);
...
... //do something to display the content item
}
```

BVI_KMProgramManager::getDefaultBookmarkedContentList()

Retrieves the list of default bookmarked content for the specified visitor.

JavaScript

```
BVI_ValueList getDefaultBookmarkedContentList(
    string serviceId,
    long userId );
```

Java

```
public final BVI_ValueList getDefaultBookmarkedContentList(
    String serviceId,
    long userId)
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	A string; the ID or name of the current service.
userId	An integer; user ID for the current visitor.

Return value

A list of BVI_Value objects, each containing a BVI_Properties object.

Remarks

To extract the BVI_Properties objects for use in a script, loop through the list using the BVI_ValueList.get method and the BVI_Value.objectValue attribute. The BVI_Properties objects contain three long values: OID, CONTENT_TYPE, and SERVICE_ID. Pass these three values to BVI_ContentManager.contentByOID() to retrieve the actual content.

The retrieved list of default bookmarked content is based on the visitor's user template. (See "User profile and related tables" on page 291.)

See also

getBookmarkedContentList()().

BVI_KMProgramManager::isBookmarkedContent()

Checks whether a content item is bookmarked for the specified visitor.

JavaScript

```
boolean isBookmarkedContent(
    string serviceId,
    long userId,
    string contentType,
    long contentOID );
```

Java

```
public final boolean isBookmarkedContent(
    String serviceId,
    long userId,
    String contentType,
    long contentOID)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`userId`	An integer; user ID for the current visitor.
`contentType`	ID or name of the content type.
`contentOID`	Content OID.

Return value — True if the content is bookmarked for the specified visitor; false otherwise.

BVI_KMProgramManager::isChannelVisible()

Checks whether a channel is visible to the specified visitor (`userId`).

JavaScript

```
boolean isChannelVisible(
    string serviceId,
    string channelId,
    long userId);
```

Java

```
public final boolean isChannelVisible(
    String serviceId,
    String channelId,
    long userId)
```

Java exception — Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`channelId`	A string; the ID or full path name of the channel.
`userId`	An integer; user ID for the current visitor.

Return value — True or false, depending on site and profile settings.

Remarks — This method returns true if the InfoExchange Portal parameter `ep_enable_qualifiers` in `bv1to1.conf` is set to zero. A zero setting for `ep_enable_qualifiers` means that qualifiers are disabled in this site and that makes all channels visible to all visitors.

If `ep_enable_qualifiers` is set to one, this method compares the channel qualifier settings to the qualifier settings for the visitor (`userId`). If the qualifiers do not filter out the `channelId` for this `userId`, this method returns true; false if the `channelId` is filtered out.

BVI_KMProgramManager::isProgramVisible()

Checks whether a program is visible to the specified visitor (`userId`).

JavaScript

```
boolean isProgramVisible(
    string serviceId,
    string programId,
    long userId);
```

Java

```
public final boolean isProgramVisible(
    String serviceId,
    String programId,
    long userId)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string; the ID or name of the current service.
`programId`	A string; the ID or full path name of the program.
`userId`	An integer; user ID for the current visitor.

Return value

True or false, depending on site and profile settings.

Remarks

This method returns true if the InfoExchange Portal parameter `ep_enable_qualifiers` in `bv1to1.conf` is set to zero. A zero setting for `ep_enable_qualifiers` means that qualifiers are disabled in this site and that makes all programs visible to all visitors.

If `ep_enable_qualifiers` is set to "1", this method compares the program qualifier settings to the qualifier settings for the visitor (`userId`). If the qualifiers do not filter out the `programId` for this `userId`, then this method returns true; false if the `programId` is filtered out.

BVI_KMProgram

This interface provides access to the underlying data type that supports programs. It is a read-only interface. These calls return information highly useful in displaying program items or creating links to programs.

This component does not allow dynamic properties.

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; Java classes reside in the `com.broadvision.ieportal.components` Java packages

This section discusses the following:

BVI_KMProgram attributes

The following table lists the attributes for BVI_KMProgram.

Attribute	Functionality
programId	Gets the program ID for this BVI_KMProgram object.
programName	Gets the program name for this BVI_KMProgram object.
programDescription	Gets the program description.
programPageType	Gets the page type for the program.
programPagePath	Gets the page path or URL for the program represented by this BVI_KMProgram object.
programType	Gets the program type.
programMgrName	Gets the name of the manager for the program represented by this BVI_KMProgram object.
programMgrEmail	Gets the email address of the manager for the program represented by this BVI_KMProgram object.
programSrcType	Gets the type of the source for this program.
programSrcId	Gets the ID of the source for the program.
programSrcContentType	Gets the content type for the content accessed by this program.
includeSubcategories	Tells whether subcategories are included in content retrieval for this program.
showOnlyReadable	Tells whether this BVI_KMProgram object is set to show only the content links for which visitors have read access.
embedScriptPath	A string containing the path to an embedded script, which is used to display a program in a home page block.
intcol1	First extra numeric data field.
intcol2	Second extra numeric data field.
intcol3	Third extra numeric data field.
strcol1	First extra text data field.
strcol2	Second extra text data field.
strcol3	First extra text data field.

BVI_KMProgram::programId

Gets the program ID for this `BVI_KMProgram` object.

JavaScript

```
readonly attribute long programId;
```

Java

```
public final long getProgramId()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value An integer representing the program ID.

BVI_KMProgram::programName

Gets the program name for this `BVI_KMProgram` object.

JavaScript
```
readonly attribute string programName;
```

Java
```
public final String getProgramName()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value A string containing the program name.

BVI_KMProgram::programDescription

Gets the program description.

JavaScript
```
readonly attribute string programDescription;
```

Java
```
public final String getProgramDescription()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value A string containing the program's description.

BVI_KMProgram::programPageType

Gets the page type for the program.

JavaScript
```
readonly attribute long programPageType;
```

Java
```
public final long getProgramPageType()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value An integer representing the page type; either 0 or 1.

Remarks The value of this attribute indicates what type of program this `BVI_KMProgram` object represents. A program can either display a page script, go to a URL, or call a portlet.

Value	Page type
0	Script
1	URL
2	Portlet

BVI_KMProgram::programPagePath

Gets the page path or URL for the program represented by this BVI_KMProgram object.

JavaScript
```
readonly attribute string programPagePath;
```

Java
```
public final String getProgramPagePath()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value A string representing the page path or URL depending on the page type of the program.

Remarks When using this attribute to build a link, test the `programPageType` attribute to determine how to construct the URL.

Example
```
ptlt_item = ptlt_programList.get(i);
    ptlt_program = ptlt_item.objectValue;
    ptlt_nextPage = ptlt_program.programPagePath;
    var ptlt_pageType = '';
    if( ptlt_program.programPageType == ptlt_URL_Type )
        ptlt_pageType= ptlt_nextPage; // call a URL
    else if( ptlt_program.programPageType == ptlt_SCRIPT_Type )
        ptlt_pageType= fastconcat(makeScriptURL(ptlt_programPage),
         "&programId=", ptlt_program.programId, "&programPage=",
         ptlt_nextPage); // call a script
    else{ // if program contains a portlet
        var progPtlt = ptltMgr.getPortlet(0, ptlt_userID,
         ptlt_program.programId);
        ptlt_pageType = fastconcat( makeScriptURL(ptlt_programPage),
            "&programId=", program.programId,
            "&programPage=", progPtlt.get("PTLT_PAGE_SCPT"));
```

BVI_KMProgram::programType

Gets the program type.

JavaScript
```
readonly attribute long programType;
```

Java `public final long getProgramType()`

Java exception Throws BVWComponentException.

Parameters None.

Return value An integer representing the program type. The program type is a content item in the BV_KM_PROGRAM_TYPE table and defines the content type and rule set category used by the program. See Appendix A, "Database schema" for further information.

BVI_KMProgram::programMgrName

Gets the name of the manager for the program represented by this BVI_KMProgram object.

JavaScript `readonly attribute string programMgrName;`

Java `public final String getProgramMgrName()`

Java exception Throws BVWComponentException.

Parameters None.

Return value A string, containing the name of the program's manager.

BVI_KMProgram::programMgrEmail

Gets the email address of the manager for the program represented by this BVI_KMProgram object.

JavaScript `readonly attribute string programMgrEmail;`

Java `public final String getProgramMgrEmail()`

Java exception Throws BVWComponentException.

Parameters None.

Return value A string, containing the program manager's email address.

BVI_KMProgram::programSrcType

Gets the type of the source for this program.

JavaScript `readonly attribute long programSrcType;`

Java `public final long getProgramSrcType()`

Java exception Throws BVWComponentException.

Parameters None.

Return value An integer representing the source type. This attribute is only used if the program's `programPageType` is 0 (script).

Remarks The value of this attribute is an integer.

Value	Source type	Description
0	Rule Set	program is defined with a rule set
1	Category	program is defined with a category
2	Content	program is defined with a content item
3	Script	program simply executes a script. This is included for backward compatibility only.
4	URL	program points to an external URL.
5	Portlet	program calls a portlet.

BVI_KMProgram::programSrcId

Gets the ID of the program's source.

JavaScript
```
readonly attribute long programSrcId;
```

Java
```
public final long getProgramSrcId()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value An integer representing the ID of the program's source. This attribute is only used if the programPageType for the program is 0 (script).

Remarks The ID for the rule set, the ID for the category, or the ID for the content item.

BVI_KMProgram::programSrcContentType

Gets the content type for the content accessed by this program.

JavaScript
```
readonly attribute long programSrcContentType;
```

Java
```
public final long getProgramSrcContentType()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value An integer representing the content type. This attribute is only used if the `programPageType` for the program is 0 (script)

Remarks The content type is retrieved based on the value of the BVI_KMProgram::programType attribute.

BVI_KMProgram::includeSubcategories

Tells whether subcategories are included in content retrieval for this program.

JavaScript
```
readonly attribute long includeSubcategories;
```

Java
```
public final long getIncludeSubcategories()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value True if subcategories are included. This attribute is only used if the `programPageType` for the program is 1 (category).

BVI_KMProgram::showOnlyReadable

Tells whether this BVI_KMProgram object is set to show only the content links for which visitors have read access.

JavaScript
```
readonly attribute long showOnlyReadable;
```

Java
```
public final long getShowOnlyReadable()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value True if the InfoExchange Portal Administration tool has set this attribute for the associated program object.

Remarks If this attribute returns true, call BVI_KMProgramManager::isContentReadable() to determine whether the visitor has read access to the content represented by this BVI_KMProgram object before displaying a link to the content.

BVI_KMProgram::embedScriptPath

Returns a string containing the path to an embedded script, which is used to display a program in a home page block.

JavaScript `readonly attribute string embedScriptPath;`

Java `public final String getEmbedScriptPath()`

Java exception Throws BVWComponentException.

Remarks To display a program in the home page, give the program a block script. This script can display any data. The site administrator can then include this program inside a block.

BVI_KMProgram::intcol1

First extra numeric data field.

JavaScript `readonly attribute long intcol1;`

Java `public final long getIntcol1()`

Java exception Throws BVWComponentException.

BVI_KMProgram::intcol2

Second extra numeric data field.

JavaScript `readonly attribute long intcol2;`

Java `public final long getIntcol2()`

Java exception Throws BVWComponentException.

BVI_KMProgram::intcol3

Third extra numeric data field.

JavaScript `readonly attribute long intcol3;`

Java `public final long getIntcol3()`

Java exception Throws BVWComponentException.

BVI_KMProgram::strcol1

First extra text data field.

JavaScript `readonly attribute string strcol1;`

Java `public final String getStrcol1()`

Java exception Throws BVWComponentException.

BVI_KMProgram::strcol2

Second extra text data field.

JavaScript `readonly attribute string strcol2;`

Java `public final String getStrcol2()`

Java exception Throws BVWComponentException.

BVI_KMProgram::strcol3

Third extra text data field.

JavaScript `readonly attribute string strcol3;`

Java `public final String getStrcol3()`

Java exception Throws BVWComponentException.

BVI_KMProgram methods

The following table lists the methods for BVI_KMProgram.

Method	Functionality
creator()	Creates a new BVI_KMProgram object.
creator()	Creates a copy of the specified BVI_KMProgram object.
clone()	Creates a copy of this BVI_KMProgram object (JavaScript).
clone_J()	Creates a copy of this BVI_KMProgram object (Java).
setBookmark()	Sets a bookmark for the specified visitor on the program represented by this BVI_KMProgram object.
isBookmarked()	Checks whether the program represented by this BVI_KMProgram object is bookmarked for the specified visitor.
getContentList()	Retrieves the content list for the program represented by this BVI_KMProgram object.
isVisible()	Checks whether this program is visible to the specified visitor.

BVI_KMProgram::creator()

Creates a new BVI_KMProgram object.

JavaScript
```
BVI_KMProgram creator();
```

Java
```
public BVI_KMProgram creator()
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters None.

Return value A new BVI_KMProgram object; in JavaScript returns null if an error occurs.

BVI_KMProgram::creator()

Creates a copy of the specified BVI_KMProgram object.

JavaScript
```
BVI_KMProgram creator(BVI_KMProgram ref);
```

Java
```
public BVI_KMProgram creator(BVI_KMProgram ref)
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

`ref`	Reference to an existing BVI_KMProgram object.

Return value Returns a BVI_KMProgram object; in JavaScript returns null if an error occurs.

BVI_KMProgram::clone()

Creates a copy of this BVI_KMProgram object.

JavaScript
```
BVI_KMProgram clone();
```

Parameters None.

Return value Returns a BVI_KMProgram object; returns null if an error occurs.

BVI_KMProgram::clone_J()

Creates a copy of this BVI_KMProgram object.

Java `public BVI_KMProgram clone_J()`

Java exception Throws BVWComponentException.

Parameters None.

Return value Returns a BVI_KMProgram object.

BVI_KMProgram::setBookmark()

Sets a bookmark for the specified visitor on the program represented by this BVI_KMProgram object.

JavaScript `void setBookmark( long userId, boolean flag );`

Java `public final void setBookmark(long userId, boolean flag)`

Java exception Throws BVWComponentException.

Parameters

`userId`	An integer; the ID for the current visitor.
`flag`	A boolean; set this true to place a bookmark on this program, false to remove a bookmark.

Return value None.

BVI_KMProgram::isBookmarked()

Checks whether the program represented by this BVI_KMProgram object is bookmarked for the specified visitor.

JavaScript `boolean isBookmarked(long userId);`

Java `public final boolean isBookmarked(long userId)`

Java exception Throws BVWComponentException.

Parameters

`userId`	An integer; the ID for the current visitor.

Return value True if the program is bookmarked for the specified visitor; false otherwise.

BVI_KMProgram::getContentList()

Retrieves the content list for the program represented by this BVI_KMProgram object.

JavaScript

```
BVI_ContentList getContentList(
    long user_id,
    BVI_SessnProf sprof,
    long max_count );
```

Java

```
public final BVI_ContentList getContentList(
    long user_id,
    BVI_SessnProf sprof,
    long max_count)
```

Java exception

Throws BVWComponentException.

Parameters

`userId`	An integer; the ID for the current visitor.
`sprof`	Session profile for this visitor.
`maxCount`	An integer; the maximum count of programs to return

Return value

A list of BVI_Content objects.

BVI_KMProgram::isVisible()

Checks whether this program is visible to the specified visitor.

JavaScript

```
boolean isVisible(long userId);
```

Java

```
public final boolean isVisible(long userId)
```

Java exception

Throws BVWComponentException.

Parameters

`userId`	An integer; user ID for the current visitor.

Return value

True if the program is not filtered out by qualifier settings.

Remarks

This method returns true if the InfoExchange Portal parameter `ep_enable_qualifiers` in `bv1to1.conf` is set to zero. A zero setting for `ep_enable_qualifiers` means qualifiers are disabled in this site and that makes all programs visible to all visitors.

If `ep_enable_qualifiers` is set to one, this method compares the program qualifier settings to the qualifier settings for the visitor (`userId`). If the qualifiers do not filter out this program for this `userId`, this method returns true; returns false otherwise.

BVI_KMChannel

This interface provides access to the underlying data type that supports channels. It is a read-only interface.

This interface does not support dynamic properties.

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; Java classes reside in the `com.broadvision.ieportal.components` Java package.

BVI_KMChannel attributes

The following table lists the attributes set by BVI_KMChannel.

Attribute	Functionality
channelId	Gets the channel OID for this BVI_KMChannel object.
parentId	Gets the parent channel ID of this BVI_KMChannel object.
channelName	Gets the channel name for this BVI_KMChannel object.
channelDescription	Gets the channel description for this BVI_KMChannel object.
channelPageType	Gets the internal page type for this BVI_KMChannel object.
channelPagePath	Gets the path to the page or URL this BVI_KMChannel object represents.
channelMgrName	Gets the name of the manager of this channel.
channelMgrEmail	Gets the e-mail address of the manager of this channel.

BVI_KMChannel::channelId

Gets the channel OID for this BVI_KMChannel object.

JavaScript

```
readonly attribute long channelId;
```

Java

```
public final long getChannelId()
```

Java exception

Throws BVWComponentException.

Return value

A long integer representing the channel OID.

BVI_KMChannel::parentId

Gets the parent channel ID of this BVI_KMChannel object.

JavaScript	`readonly attribute long parentId;`
Java	`public final long getParentId()`
Java exception	Throws BVWComponentException.
Return value	An integer representing the OID of the parent channel.

BVI_KMChannel::channelName

Gets the channel name for this BVI_KMChannel object.

JavaScript	`readonly attribute string channelName;`
Java	`public final String getChannelName()`
Java exception	Throws BVWComponentException.
Return value	A string, containing the channel name.

BVI_KMChannel::channelDescription

Gets the channel description for this BVI_KMChannel object.

JavaScript	`readonly attribute string channelDescription;`
Java	`public final String getChannelDescription()`
Java exception	Throws BVWComponentException.
Return value	A string containing the channel description.
Remarks	This description is set by the InfoExchange Portal Admin Tool.

BVI_KMChannel::channelPageType

Gets the internal page type for this BVI_KMChannel object.

JavaScript
```
readonly attribute long channelPageType;
```

Java
```
public final long getChannelPageType()
```

Java exception Throws BVWComponentException.

Return value An integer representing the page type; either 0 or 1. See remarks.

Remarks The value of this attribute indicates what type of channel this BVI_KMChannel object represents. A channel can either display a page script or go to a URL.

Value	Page type
0	Script
1	URL

BVI_KMChannel::channelPagePath

Gets the path to the page or URL this BVI_KMChannel object represents.

JavaScript
```
readonly attribute string channelPagePath;
```

Java
```
public final String getChannelPagePath()
```

Java exception Throws BVWComponentException.

Return value A string, containing the path to the page or URL.

Remarks The string contains either a URL or a script pathname, depending on the `channelPageType`.

BVI_KMChannel::channelMgrName

Gets the name of the manager of this channel.

JavaScript
```
readonly attribute string channelMgrName;
```

Java
```
public final String getChannelMgrName()
```

Java exception Throws BVWComponentException.

Return value A string, containing the name of the manager of this channel.

BVI_KMChannel::channelMgrEmail

Gets the e-mail address of the manager of this channel.

JavaScript
```
readonly attribute string channelMgrEmail;
```

Java
```
public final String getChannelMgrEmail()
```

Java exception Throws BVWComponentException.

Return value A string, containing the e-mail address of the manager of this channel.

BVI_KMChannel methods

The following table lists the methods set by BVI_KMChannel.

Method	Functionality
creator()	Creates a new empty BVI_KMChannel object.
creator()	Creates a copy of the specified BVI_KMChannel object.
clone()	Creates a copy of this BVI_KMChannel object (JavaScript).
clone_J()	Creates a copy of this BVI_KMChannel object (Java).
setBookmark()	Sets or removes a bookmark on the channel represented by this BVI_KMChannel object.
isBookmarked()	Checks whether this channel is bookmarked for the specified visitor.
isVisible()	Checks whether the channel is visible to the specified visitor.

BVI_KMChannel::creator()

Creates a new empty BVI_KMChannel object.

JavaScript
```
BVI_KMChannel creator()
```

Java
```
public BVI_KMChannel creator()
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Return value A BVI_KMChannel object; in JavaScript returns null if an error occurs.

BVI_KMChannel::creator()

Creates a copy of the specified BVI_KMChannel object.

JavaScript
```
BVI_KMChannel creator(BVI_KMChannel ref)
```

Java
```
public BVI_KMChannel creator(BVI_KMChannel ref)
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

`ref`	Reference to an existing BVI_KMChannel object.

Return value A BVI_KMChannel object; in JavaScript returns null if an error occurs.

BVI_KMChannel::clone()

Creates a copy of this BVI_KMChannel object.

JavaScript
```
BVI_KMChannel clone();
```

Return value A BVI_KMChannel object; in returns null if an error occurs.

BVI_KMChannel::clone_J()

Creates a copy of this BVI_KMChannel object.

Java
```
public BVI_KMChannel clone_J()
```

Java exception Throws BVWComponentException.

Return value A BVI_KMChannel object.

BVI_KMChannel::setBookmark()

Sets or removes a bookmark on the channel represented by this BVI_KMChannel object.

JavaScript
```
void setBookmark(long userId, boolean flag);
```

Java
```
public final void setBookmark(long userId, boolean flag)
```

Java exception Throws BVWComponentException.

Parameters

`userId`	An integer; user ID for the current visitor.
`flag`	Set this true to place a bookmark on this channel; false to remove a bookmark.

Return value None.

BVI_KMChannel::isBookmarked()

Checks whether this channel is bookmarked for the specified visitor.

JavaScript
```
boolean isBookmarked(long userId);
```

Java
```
public final boolean isBookmarked(long userId)
```

Java exception Throws BVWComponentException.

Parameters

`userId`	ID for the current visitor.

Return value True if the channel is bookmarked for the specified visitor; false otherwise.

BVI_KMChannel::isVisible()

Checks whether the channel is visible to the specified visitor.

JavaScript
```
boolean isVisible(long userId);
```

Java
```
public final boolean isVisible(long userId)
```

Java exception Throws BVWComponentException.

Parameters

`userId`	ID for the current visitor.

Return value True if the channel is not filtered out by qualifier settings.

Remarks This method returns true if the InfoExchange Portal parameter `ep_enable_qualifiers` in `bv1to1.conf` is set to zero. A zero setting for `ep_enable_qualifiers` means that qualifiers are disabled in this site and that makes all channels visible to all visitors.

If `ep_enable_qualifiers` is set to one, this method compares the channel qualifier settings to the qualifier settings for the visitor (`userId`). If the qualifiers do not filter out this channel for this `userId`, then this method returns true; false otherwise.

BVI_EPFilterManager

This interface provides the means to manipulate qualifiers.

This component does not allow dynamic properties.

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; Java classes reside in the `com.broadvision.ieportal.components` Java package.

This section contains information about the following methods which define the valid blockType values used by the BVI_EPFilterManager class:

Method	Functionality
creator()	Creates a new BVI_EPFilterManager object.
creator()	Creates a copy of a BVI_EPFilterManager object.
clone()	Creates a copy of this BVI_EPFilterManager object (JavaScript).
clone_J()	Creates a copy of this BVI_EPFilterManager object (Java).
Administrative methods	
addQualifier()	Creates a new qualifier in a service.
renameQualifier()	Renames a qualifier in a service.
deleteQualifier()	Deletes a qualifier from a service
deleteQualifiers()	Deletes a list of qualifiers from a service.
addQualVal()	Adds a qualifier value to a service.
updateQualVal()	Changes a qualifier value in a service.
deleteQualVal()	Deletes a qualifier value in a service.
deleteQualVals()	Deletes a list of qualifier values in a service.
Tagging methods	
setQualValsForContent()	Sets a list of qualifier values on a program, user template, or content object in a service.
unsetQualValsForContent()	Clears a list of qualifier values on a program, user template, or content object in a service.
unsetQualifierForContent()	Clears all values of a qualifier for a program, user template, or content object in a service.
replaceQualValsForContent()	Replaces a list of qualifier values on a program, user template, or content object in a service.
setQualValsForChannel()	Sets the qualifier values for a channel in a service.
unsetQualValsForChannel()	Clears the qualifier values for a channel in a service.
unsetQualifierForChannel()	Clears all the values of a qualifier for a channel in a service.
replaceQualValsForChannel()	Replaces the qualifier values for a channel in a service.
setQualValsForUser()	Sets a list of qualifier values for a visitor.
unsetQualValsForUser()	Clears a list of qualifier values for a visitor.
unsetQualifierForUser()	Clears all values of a qualifier for a visitor.
replaceQualValsForUser()	Replaces a list of qualifier values for a visitor.
replaceSettableQualValsForUser()	Replaces a list of visitor-settable qualifier values in the current service for the specified visitor.

Method	Functionality
Reporting methods	
getQualifierList()	Gets the list of qualifiers in a service.
getQualifierList ByLetter()	Gets the list of qualifiers in a service that start with the specified letter.
getQualVals()	Gets the list of qualifiers in a service.
getSettableQualVals()	Gets a list of visitor-settable qualifier values in a service.
getQualValsOnContent()	Gets the qualifier values set on a content object in a service.
getQualValsOnChannel()	Gets the qualifier values set on a channel in a service.
getQualValsForUser()	Gets the qualifier values set for a visitor in a service.
getSettableQualValsForUser()	Gets a list of the qualifier values a given visitor can set in the current service.
Filtering methods	
isContentVisible()	Checks to see whether a content item can be seen by the visitor.
Filter cache administration methods	
setFilterCacheSize()	Sets the size of the filter cache.
getFilterCacheSize()	Gets the size of the filter cache.
getFilterCacheEntryNum()	Returns the number of items in the cache for a given service.
getFilteringStatus()	Determines whether the filter cache is enabled or disabled.
getFilterEnabledContentTypes()	Gets the list of content types to be filtered based on the visitor's qualifiers.
Filter cache warmup/dump methods	
dumpcache()	Dumps the OID list for the specified store ID and content type.
cachewarmup()	Pre-loads a specific content cache with content specified by an OID list.

BVI_EPFilterManager::creator()

Creates a new BVI_EPFilterManager object.

JavaScript

```
BVI_EPFilterManager creator(long serviceId, long userId);
```

Java

```
public BVI_EPFilterManager creator(long serviceId, long userId);
```

Java exception

Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

`serviceId`	An integer containing the service ID
`userId`	An integer containing the ID for the current user account

Return value

A BVI_EPFilterManager object; in JavaScript returns null if an error occurs.

BVI_EPFilterManager::creator()

Creates a copy of a BVI_EPFilterManager object.

JavaScript
```
BVI_EPFilterManager creator(BVI_EPFilterManager ref)
```

Java
```
public BVI_EPFilterManager creator(BVI_EPFilterManager ref)
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

`ref`	An existing BVI_EPFilterManager object.

Return value A BVI_EPFilterManager object; in JavaScript returns null if an error occurs.

BVI_EPFilterManager::clone()

Creates a copy of this BVI_EPFilterManager object.

JavaScript
```
BVI_EPFilterManager clone()
```

Parameters None.

Return value A copy of the BVI_EPFilterManager object; returns null if an error occurs.

BVI_EPFilterManager::clone_J()

Creates a copy of this BVI_EPFilterManager object.

Java
```
public final BVI_EPFilterManager clone_J()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value A copy of the BVI_EPFilterManager object.

Administrative methods

The following methods manipulate qualifiers and their values in a service.

Method	Functionality
addQualifier()	Creates a new qualifier in a service.
renameQualifier()	Renames a qualifier in a service.
deleteQualifier()	Deletes a qualifier from a service
deleteQualifiers()	Deletes a list of qualifiers from a service.
addQualVal()	Adds a qualifier value to a service.
updateQualVal()	Changes a qualifier value in a service.
deleteQualVal()	Deletes a qualifier value in a service.
deleteQualVals()	Deletes a list of qualifier values in a service.

BVI_EPFilterManager::addQualifier()

Creates a new qualifier in a service.

JavaScript
```
long addQualifier(string qualName);
```

Java
```
public final long addQualifier(string qualName);
```

Java exception Throws BVWComponentException.

Parameters

qualName	Qualifier name.

Return value If successful, returns the ID of the newly created qualifier, otherwise returns zero.

BVI_EPFilterManager::renameQualifier()

Renames a qualifier in a service.

JavaScript
```
renameQualifier(long qualifierId, string newQualName);
```

Java
```
public final renameQualifier(long qualifierId, string newQualName);
```

Java exception Throws BVWComponentException.

Parameters

qualifierId	ID for the qualifier.
newQualName	New name for the qualifier.

Return value None.

BVI_EPFilterManager::deleteQualifier()

Deletes a qualifier from a service. All values of and references to the deleted qualifiers are also deleted from the service.

JavaScript
```
void deleteQualifier( long qualifierId );
```

Java
```
public final void deleteQualifier( long qualifierId );
```

Java exception Throws BVWComponentException.

Parameters

qualifierId	ID for the qualifier.

Return value None.

BVI_EPFilterManager::deleteQualifiers()

Deletes a list of qualifiers from a service. All values of and references to the deleted qualifiers are also deleted from the service.

JavaScript
```
void deleteQualifiers( BVI_ValueList qualifierIdList );
```

Java
```
public final void deleteQualifiers( BVI_ValueList qualifierIdList );
```

Java exception Throws BVWComponentException.

Parameters

qualifierIdList	ID list of qualifiers.

Return value None.

BVI_EPFilterManager::addQualVal()

Adds a qualifier value to a service.

JavaScript
```
long addQualVal(
    long qualifierId,
    string qualValName,
    boolean userSettable );
```

Java

```
public final long addQualVal(
    long qualifierId,
    String qualValName,
    boolean userSettable );
```

Java exception Throws BVWComponentException.

Parameters

`qualifierId`	ID for the qualifier.
`qualValName`	Name for the qualifier value.
`userSettable`	Determines whether the qualifier value can be set by visitors.

Return value On success, the ID of the newly created qualifier value; otherwise 0 on failure.

BVI_EPFilterManager::updateQualVal()

Changes a qualifier value in a service.

JavaScript

```
void updateQualVal(
    long qualifierId,
    long qualValId,
    string newQualValName,
    boolean userSettable );
```

Java

```
public final void updateQualVal(
    long qualifierId,
    long qualValId,
    String newQualValName,
    boolean userSettable );
```

Java exception Throws BVWComponentException.

Parameters

`qualifierId`	ID for the qualifier.
`qualValId`	ID for the qualifier value.
`newQualValName`	A new name for the qualifier value.
`userSettable`	Determines whether the qualifier value can be set by visitors.

Return value None.

BVI_EPFilterManager::deleteQualVal()

Deletes a qualifier value in a service.

JavaScript

```
void deleteQualVal( long qualifierId, long qualValId );
```

Java
```
public final void deleteQualVal( long qualifierId, long qualValId );
```

Java exception Throws BVWComponentException.

Parameters

qualifierId	ID for the qualifier.
qualValId	ID for the qualifier value.

Return value None.

BVI_EPFilterManager::deleteQualVals()

Deletes a list of qualifier values in a service.

JavaScript
```
void deleteQualVals(
    long qualifierId,
    BVI_ValueList qualValIdList );
```

Java
```
public final void deleteQualVals(
    long qualifierId,
    BVI_ValueList qualValIdList );
```

Java exception Throws BVWComponentException.

Parameters

qualifierId	ID for the qualifier.
qualValIdList	ID list of qualifier values.

Return value None.

Tagging content methods

The following methods enable you to set, get, change, and remove qualifiers and qualifier values on different objects.

Method	Functionality
setQualValsForContent()	Sets a list of qualifier values on a program, user template, or content object in a service.
unsetQualValsForContent()	Clears a list of qualifier values on a program, user template, or content object in a service.
unsetQualifierForContent()	Clears a qualifier value for a program, user template, or content object in a service.
replaceQualValsForContent()	Replaces a list of qualifier values on a program, user template, or content object in a service.
setQualValsForChannel()	Sets the qualifier values for a channel in a service.

Method	Functionality
unsetQualValsForChannel()	Clears the qualifier values for a channel in a service.
unsetQualifierForChannel()	Clears all values of a qualifier for a channel in a service.
replaceQualValsForChannel()	Replaces the qualifier values for a channel in a service.
setQualValsForUser()	Sets a list of qualifier values for a visitor.
unsetQualValsForUser()	Clears a list of qualifier values for a visitor.
unsetQualifierForUser()	Clear a qualifier value for a visitor.
replaceQualValsForUser()	Replaces a list of qualifier values for a visitor.
replaceSettableQualValsForUser()	Replaces a list of visitor-settable qualifier values in the current service for the specified visitor.

BVI_EPFilterManager::setQualValsForContent()

Sets a list of qualifier values on a program, user template, or content object in a service. This method can be used for setting qualifier values on any content object including program, user template, and normal content item, providing an incremental way of tagging.

JavaScript

```
void setQualValsForContent(
    long contentTypeId,
    long contentOID,
    BVI_ValueList qualValList );
```

Java

```
public final void setQualValsForContent(
    long contentTypeId,
    long contentOID,
    BVI_ValueList qualValList );
```

Java exception

Throws BVWComponentException.

Parameters

`contentTypeId`	ID of content type.
`contentOID`	ID of the content item.

`qualValList`	A BVI_ValueList containing pairs of a qualifier ID and qualifier value ID.
	
	`(QID, QVID)`
	`(QID, QVID)`
	`…`
	`(QID, QVID)`
	
	The qualValList is a BVI_ValueList. Each BVI_Value of the list is a BVI_ValueList with a length of 2. The first BVI_Value is a qualifier ID, the second is a qualifier value ID.

Return value

None.

BVI_EPFilterManager::unsetQualValsForContent()

Clears a list of qualifier values on a program, user template, or content object in a service.

JavaScript

```
void unsetQualValsForContent(
    long contentTypeId,
    long contentOID,
    BVI_ValueList qualValList );
```

Java

```
public final void unsetQualValsForContent(
    long contentTypeId,
    long contentOID,
    BVI_ValueList qualValList );
```

Java exception

Throws BVWComponentException.

Parameters

contentTypeId	ID of content type.
contentOID	ID of the content item.
qualValList	A BVI_ValueList containing a qualifier ID and qualifier value ID.

Return value

None.

BVI_EPFilterManager::unsetQualifierForContent()

Clears all values of a qualifier for a program, user template, or content object in a service.

JavaScript

```
void unsetQualifierForContent(
    long contentTypeId,
    long contentOID,
    long qualifierId);
```

Java

```
public final void unsetQualifierForContent(
    long contentTypeId,
    long contentOID,
    long qualifierId);
```

Java exception

Throws BVWComponentException.

Parameters

contentTypeId	ID of content type.
contentOID	ID of the content item.
qualifierId	ID for the qualifier.

BVI_EPFilterManager::replaceQualValsForContent()

Replaces a list of qualifier values on a content object in a service. This method can be used for setting qualifier values on any content object including program, user template, and normal content item.

JavaScript

```
void replaceQualValsForContent(
    long contentTypeId,
    long contentOID,
    BVI_ValueList qualValList );
```

Java

```
public final void replaceQualValsForContent(
    long contentTypeId,
    long contentOID,
    BVI_ValueList qualValList );
```

Java exception

Throws BVWComponentException.

Parameters

`contentTypeId`	ID of content type.
`contentOID`	ID of the content item.
`qualValList`	A BVI_ValueList containing a qualifier ID and qualifier value ID.

Return value

None.

BVI_EPFilterManager::setQualValsForChannel()

Sets the qualifier values for a channel in a service.

JavaScript

```
void setQualValsForChannel(
    long channelId,
    BVI_ValueList qualValList,
    boolean recursiveFlag );
```

Java

```
public final void setQualValsForChannel(
    long channelId,
    BVI_ValueList qualValList,
    boolean recursiveFlag );
```

Java exception

Throws BVWComponentException.

Parameters

`channelId`	ID for the channel.

`qualValList`	A BVI_ValueList containing pairs of a qualifier ID and qualifier value ID.
	
	`(QID, QVID)`
	`(QID, QVID)`
	`…`
	`(QID, QVID)`
	
	The qualValList is a BVI_ValueList. Each BVI_Value of the list is a BVI_ValueList with a length of 2. The first BVI_Value is a qualifier ID, the second is a qualifier value ID.

`recursiveFlag`	Determines whether the qualifier values are also set for sub channels of the specified channel.

Return value None.

BVI_EPFilterManager::unsetQualValsForChannel()

Clears the qualifier values for a channel in a service.

JavaScript

```
void unsetQualValsForChannel(
    long channelId,
    BVI_ValueList qualValList,
    boolean recursiveFlag );
```

Java

```
public final void unsetQualValsForChannel(
    long channelId,
    BVI_ValueList qualValList,
    boolean recursiveFlag );
```

Java exception Throws BVWComponentException.

Parameters

`channelId`	ID for the channel.

`qualValList`	A BVI_ValueList containing pairs of a qualifier ID and qualifier value ID.
	
	`(QID, QVID)`
	`(QID, QVID)`
	`…`
	`(QID, QVID)`
	
	The qualValList is a BVI_ValueList. Each BVI_Value of the list is a BVI_ValueList with a length of 2. The first BVI_Value is a qualifier ID, the second is a qualifier value ID.

`recursiveFlag`	Determines whether the qualifier values are also set for sub channels of the specified channel.

Return value None.

BVI_EPFilterManager::unsetQualifierForChannel()

Clears all values of a qualifier for a channel in a service.

JavaScript

```
void unsetQualifierForChannel(
    long channelId,
    long qualifierId,
    boolean recursiveFlag);
```

Java

```
public final void unsetQualifierForChannel(
    long channelId,
    long qualifierId,
    boolean recursiveFlag);
```

Java exception Throws BVWComponentException.

Parameters

`channelId`	ID for the channel.
`qualifierId`	ID for the qualifier.
`recursiveFlag`	Determines whether the qualifier values are also set for sub channels of the specified channel.

Return value None.

BVI_EPFilterManager::replaceQualValsForChannel()

Replaces the qualifier values for a channel in a service.

JavaScript

```
void replaceQualValsForChannel(
    long channelId,
    BVI_ValueList qualValList,
    boolean recursiveFlag );
```

Java

```
public final void replaceQualValsForChannel(
    long channelId,
    BVI_ValueList qualValList,
    boolean recursiveFlag );
```

Java exception

Throws BVWComponentException.

Parameters

`channelId`	ID for the channel.

`qualValList`	A BVI_ValueList containing pairs of a qualifier ID and qualifier value ID.
	
	`(QID, QVID)`
	`(QID, QVID)`
	`…`
	`(QID, QVID)`
	
	The qualValList is a BVI_ValueList. Each BVI_Value of the list is a BVI_ValueList with a length of 2. The first BVI_Value is a qualifier ID, the second is a qualifier value ID.

`recursiveFlag`	Determines whether the qualifier values are also set for sub channels of the specified channel.

Return value

None.

BVI_EPFilterManager::setQualValsForUser()

Sets a list of qualifier values for a visitor.

JavaScript

```
void setQualValsForUser(
    long userId,
    BVI_ValueList qualValList );
```

Java

```
public final void setQualValsForUser(
    long userId,
    BVI_ValueList qualValList );
```

Java exception

Throws BVWComponentException.

Parameters

`userId`	The visitor's user account ID.

`qualValList`	A BVI_ValueList containing pairs of a qualifier ID and qualifier value ID.
	
	`(QID, QVID)`
	`(QID, QVID)`
	`…`
	`(QID, QVID)`
	
	The qualValList is a BVI_ValueList. Each BVI_Value of the list is a BVI_ValueList with a length of 2. The first BVI_Value is a qualifier ID, the second is a qualifier value ID.

Return value None.

Remarks This method provides an incremental way of setting qualifier values.

BVI_EPFilterManager::unsetQualValsForUser()

Clears a list of qualifier values for a visitor.

JavaScript

```
void unsetQualValsForUser(
    long userId,
    BVI_ValueList qualValList );
```

Java

```
public final void unsetQualValsForUser(
    long userId,
    BVI_ValueList qualValList );
```

Java exception Throws BVWComponentException.

Parameters

`userId`	The visitor's user account ID.

`qualValList`	A BVI_ValueList containing pairs of a qualifier ID and qualifier value ID.
	
	`(QID, QVID)`
	`(QID, QVID)`
	`…`
	`(QID, QVID)`
	
	The qualValList is a BVI_ValueList. Each BVI_Value of the list is a BVI_ValueList with a length of 2. The first BVI_Value is a qualifier ID, the second is a qualifier value ID.

Return value None.

BVI_EPFilterManager::unsetQualifierForUser()

Clears all values of a qualifier for a visitor.

JavaScript

```
void unsetQualifierForUser( long userId, long qualifierId );
```

Java

```
public final void unsetQualifierForUser( long userId, long qualifierId );
```

Java exception

Throws BVWComponentException.

Parameters

`userId`	The visitor's user account ID.
`qualifierId`	ID for the qualifier.

Return value

None.

BVI_EPFilterManager::replaceQualValsForUser()

Replaces a list of qualifier values for a visitor.

JavaScript

```
void replaceQualValsForUser(
    long userId,
    BVI_ValueList qualValList );
```

Java

```
public final void replaceQualValsForUser(
    long userId,
    BVI_ValueList qualValList );
```

Java exception

Throws BVWComponentException.

Parameters

`userId`	The visitor's user account ID.

`qualValList`	A BVI_ValueList containing pairs of a qualifier ID and qualifier value ID.
	
	`(QID, QVID)`
	`(QID, QVID)`
	`…`
	`(QID, QVID)`
	
	The qualValList is a BVI_ValueList. Each BVI_Value of the list is a BVI_ValueList with a length of 2. The first BVI_Value is a qualifier ID, the second is a qualifier value ID.

Return value

None.

BVI_EPFilterManager::replaceSettableQualValsForUser()

Replaces a list of visitor-settable qualifier values in the current service for the specified visitor.

JavaScript

```
void replaceSettableQualValsForUser(
    long userId,
    BVI_ValueList qualValList );
```

Java

```
public final void replaceSettableQualValsForUser(
    long userId,
    BVI_ValueList qualValList );
```

Java exception

Throws BVWComponentException.

Parameters

`userId`	The visitor's user account ID.

`qualValList`	A BVI_ValueList containing pairs of a qualifier ID and qualifier value ID.
	
	`(QID, QVID)`
	`(QID, QVID)`
	`…`
	`(QID, QVID)`
	
	The qualValList is a BVI_ValueList. Each BVI_Value of the list is a BVI_ValueList with a length of 2. The first BVI_Value is a qualifier ID, the second is a qualifier value ID.

Return value

None.

Reporting methods

The following methods enable you to retrieve qualifier information for generating reports:

Method	Functionality
getQualifierList()	Gets the list of qualifiers in a service.
getQualifierList ByLetter()	Gets the list of qualifiers in a service that start with the specified letter.
getQualVals()	Gets the list of qualifiers for a visitor in a service.
getSettableQualVals()	Gets a list of visitor-settable qualifier values in the current service.
getQualValsOnContent()	Gets the qualifier values set on a program, user template, or content object in a service.
getQualValsOnChannel()	Gets the qualifier values set on a channel in a service.
getQualValsForUser()	Gets the qualifier values set for a visitor in a service.
getSettableQualValsForUser()	Gets a list of the qualifier values a given visitor can set in the current service.

BVI_EPFilterManager::getQualifierList()

Gets the list of qualifiers in a service.

JavaScript
```
BVI_ValueList getQualifierList( string nameLikeString, long maxCount );
```

Java
```
public final BVI_ValueList getQualifierList( string nameLikeString, long
maxCount );
```

Java exception Throws BVWComponentException.

Parameters

`nameLikeString`	Search string.
`maxCount`	The maximum number of qualifiers to return.

Return value A list of qualifiers containing the string in alphabetical order sorted by qualifier name. If `nameLikeString` is null, returns the first `maxCount` qualifiers for the service.

BVI_EPFilterManager::getQualifierList ByLetter()

Gets the list of qualifiers in a service that start with the specified letter.

JavaScript
```
BVI_ValueList getQualifierListByLetter( string letter);
```

Java
```
public final BVI_ValueList getQualifierListByLetter( string letter);
```

Java exception Throws BVWComponentException.

Parameters

`letter`	First letter of qualifiers to be retrieved.

Return value A list of qualifiers starting with the specified letter in alphabetical order sorted by qualifier name.

BVI_EPFilterManager::getQualVals()

Gets the list of qualifiers values in a service.

JavaScript
```
BVI_ValueList getQualVals( long qualifierId );
```

Java
```
public final BVI_ValueList getQualVals( long qualifierId );
```

Java exception Throws BVWComponentException.

Parameters

`qualifierId`	ID or name for the qualifier.

Return value A list of qualifiers in alphabetical order sorted by qualifier value.

BVI_EPFilterManager::getSettableQualVals()

Gets a list of visitor-settable qualifier values in the current service.

JavaScript
```
BVI_ValueList getSettableQualVals();
```

Java
```
public final BVI_ValueList getSettableQualVals();
```

Java exception Throws BVWComponentException.

Parameters None.

Return value A BVI_ValueList containing qualifier values that can be set by the visitor.

BVI_EPFilterManager::getQualValsOnContent()

Gets the qualifier values set on a program, user template, or content object in a service.

```
BVI_ValueList getQualValsOnContent( long contentTypeId, long contentOID
);
```

Parameters

`contentTypeId`	ID of content type.
`contentOID`	ID of the content item.

Return value A BVI_ValueList containing the qualifier values set on the specified content object. In the returned BVI_ValueList , each BVI_Value is a BVI_Properties with the following four properties:

`QID`	ID of qualifier.
`QUALIFIER`	Name of qualifier.
`QVID`	ID of qualifier value.
`VALUE`	Name of qualifier value.

BVI_EPFilterManager::getQualValsOnChannel()

Gets the qualifier values set on a channel in a service.

JavaScript
```
BVI_ValueList getQualValsOnChannel( long channelId );
```

Java
```
public final BVI_ValueList getQualValsOnChannel( long channelId );
```

Java exception
Throws BVWComponentException.

Parameters

channelId	ID for the channel.

Return value
A BVI_ValueList containing the qualifier values set on the specified channel. In the returned BVI_ValueList , each BVI_Value is a BVI_Properties with the following four properties:

QID	ID of qualifier.
QUALIFIER	Name of qualifier.
QVID	ID of qualifier value.
VALUE	Name of qualifier value.

BVI_EPFilterManager::getQualValsForUser()

Gets the qualifier values set for a visitor in a service.

JavaScript
```
BVI_ValueList getQualValsForUser( long userId );
```

Java
```
public final BVI_ValueList getQualValsForUser( long userId );
```

Java exception
Throws BVWComponentException.

Parameters

userId	The visitor's user account ID.

Return value
A BVI_ValueList containing the qualifier values set for the specified visitor.In the returned BVI_ValueList , each BVI_Value is a BVI_Properties with the following four properties:

QID	ID of qualifier.
QUALIFIER	Name of qualifier.
QVID	ID of qualifier value.
VALUE	Name of qualifier value.

BVI_EPFilterManager::getSettableQualValsForUser()

Gets a list of the qualifier values a given visitor can set in the current service.

JavaScript

```
BVI_ValueList getSettableQualValsForUser( long userId );
```

Java

```
public final BVI_ValueList getSettableQualValsForUser( long userId );
```

Java exception

Throws BVWComponentException.

Parameters

`userId`	The visitor's user account ID.

Return value

A BVI_ValueList containing the qualifier values the specified visitor can set. In the returned BVI_ValueList , each BVI_Value is a BVI_Properties with the following four properties:

`QID`	ID of qualifier.
`QUALIFIER`	Name of qualifier.
`QVID`	ID of qualifier value.
`VALUE`	Name of qualifier value.

Filtering methods

You can use the following method to verify whether a content item can be seen by a visitor.

Method	Functionality
isContentVisible()	Checks to see whether a content item can be seen by the visitor.

BVI_EPFilterManager::isContentVisible()

Checks to see whether a content item can be seen by the visitor.

JavaScript

```
boolean isContentVisible( long contentTypeId, long contentOID );
```

Java

```
public final boolean isContentVisible( long contentTypeId, long
contentOID );
```

Java exception

Throws BVWComponentException.

Parameters

`contentTypeId`	ID of content type.
`contentOID`	ID of the content item.

Return value

Returns “true” if:

- In the `bv1to1.conf` file, `ep_enable_qualifier` is set to "1" and the qualifiers on the content items match the qualifiers on the visitor.
- In the `bv1to1.conf` file, `ep_enable_qualifier` is set to "0".

Otherwise, returns “false”.

Filter cache administration methods

You can use these methods to view and change parameters related to the filter cache, which contains qualifier values for content.

Method	Functionality
setFilterCacheSize()	Sets the size of the filter cache.
getFilterCacheSize()	Gets the size of the filter cache.
getFilterCacheEntryNum()	Returns the number of items in the cache for a given service.
getFilteringStatus()	Determines whether the filter cache is enabled or disabled.
getFilterEnabledContentTypes()	Returns list of content type names that are enabled for filtering.

BVI_EPFilterManager::setFilterCacheSize()

Sets the maximum size limit of the filter cache.

JavaScript

```
void setFilterCacheSize( long size );
```

Java

```
public final void setFilterCacheSize( long size );
```

Java exception

Throws BVWComponentException.

Parameters

`size`	Maximum size limit of the filter cache. This limit is the number of entries. Each filtered object (content item, channel, program) has an entry in filter cache.

Return value

None.

BVI_EPFilterManager::getFilterCacheSize()

Gets the maximum size limit of the filter cache.

JavaScript
```
long getFilterCacheSize();
```

Java
```
public final long getFilterCacheSize();
```

Java exception Throws BVWComponentException.

Parameters None.

Return value The maximum size limit of the filter cache.

BVI_EPFilterManager::getFilterCacheEntryNum()

Returns the number of items in the cache for a given service.

JavaScript
```
long getFilterCacheEntryNum();
```

Java
```
public final long getFilterCacheEntryNum();
```

Java exception Throws BVWComponentException.

Parameters None.

Return value An integer; the number of items in the cache for the service.

BVI_EPFilterManager::getFilteringStatus()

Determines whether the filter cache is enabled or disabled.

JavaScript
```
long getFilteringStatus();
```

Java
```
public final long getFilteringStatus();
```

Java exception Throws BVWComponentException.

Parameters None.

Return value An integer: 1 for enabled, 0 for disabled.

BVI_EPFilterManager::getFilterEnabledContentTypes()

Gets the list of content types to be filtered based on the visitor's qualifiers.

JavaScript
```
BVI_StringList getFilterEnabledContentTypes();
```

Java
```
public final BVI_StringList getFilterEnabledContentTypes();
```

Java exception Throws BVWComponentException.

Parameters None.

Return value A BVI_StringList containing the filter-enabled content types defined by the configuration parameter '`ep_filter_enabled_cnt_types`'.

Filter Cache Warmup/Dump methods

You can use the following methods when you use the cache to manipulate large amounts of data.

Method	Functionality
Filter cache warmup/dump methods	
dumpcache()	Dumps the OID list for the specified store ID and content type from filter cache.
cachewarmup()	Pre-loads a filter cache with object specified by an OID list.

BVI_EPFilterManager::dumpcache()

Dumps the OID list for the specified store ID and content type from filter cache.

JavaScript
```
BVI_StringList dumpcache(string cacheid);
```

Java
```
public final BVI_StringList dumpcache(String cacheid);
```

Java exception Throws BVWComponentException.

Parameters

`cacheid` A string containing two name-value pairs specifying the store ID and content type in the structure:

`"STORE_ID=S, CONTENT_TYPE=T"`

where *S* is a valid store ID or "*" and *T* is a valid content type ID or "*".

Return value

A list of one or more strings which each contain a store ID, a content type, and a list of content OIDs separated by spaces. For example:

```
"STORE_ID=1, CONTENT_TYPE=10, OIDLIST= 1 2 4 5 100"
```

If the cache ID is set to `"STORE_ID=*, CONTENT_TYPE=*"`, this method returns all filter cache OID lists. On error, returns null and sets a flag in the Error object.

Remarks

This method is similar to the BVI_ContentManager::dumpcache method discussed in the *BroadVision One-To-One Enterprise API Reference* manual.

BVI_EPFilterManager::cachewarmup()

Pre-loads filter cache with object specified by an OID list.

JavaScript

```
long cachewarmup(string cachestr);
```

Java

```
public final long cachewarmup(String cachestr);
```

Java exception

Throws BVWComponentException.

Parameters

`cachestr`	A list of name-value pairs specifying the store ID, content type, and a list of content OIDs as acquired through dumpcache().

Return value

On success, returns 0; otherwise returns an error code.

Remarks

The cache warmup API is responsible for all the necessary data parsing. To be efficient and flexible, the cache string is used in this method to pass the data.

3 Using Organizations with InfoExchange Portal

This chapter describes the JavaScript and Java component interfaces used by InfoExchange Portal for working with organizations. *Organizations* represent a hierarchy of divisions, departments, groups, and others that are separate from qualifiers. The Site Administrator uses the Admin Tool to define the organizations for a specific site; a Service Administrator uses the Admin Tool to define the organizations for a specific service. Unlike qualifiers, a visitor who is not a Site Administrator can administer an organization.

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; the Java classes reside in the `com.broadvision.ieportal.components` Java package. These files do not follow the InfoExchange Portal naming convention because they are designed for use with other BroadVision One-To-One Enterprise applications. The interfaces described in this chapter are:

.

Interface	Functionality
BVI_MRAccount	Manages a trading account.
BVI_MRAccountManager	Performs basic account management functions.
BVI_MROrgEntityMgr	Creates and manages BVI_MROrgEntity components.
BVI_MROrgEntity	Describes organizational entities.

Except where indicated, an error condition that results from invoking a method is detected through the error component method: `Error.set`. This object is documented in the "Error" section of the "Built-in interfaces" chapter of the BroadVision One-To-One Enterprise *Application Programmer's Reference*.


For more information on these functions, see the *BroadVision Procurement Consultant's Guide*.

BVI_MRAccountManager

The `BVI_MRAccountManager` component performs the basic management functions for customer accounts and account-manager accounts.

JavaScript `$BV1TO1/include/jsi/components/mraccountmgr.jsi`

Java `com.broadvision.components.BVI_MRAccountManager.*`

This interface has the following methods:

Method	Description
newAccount()	Construct a new account object.
getAccountByName()	Get the account by its name.

BVI_MRAccountManager::newAccount()

Create a new account object.

JavaScript
```
BVI_MRAccount newAccount ( string accountname );
```

Java
```
public final BVI_MRAccount newAccount ( String accountname )
```

Java exception Throws BVWComponentException.

Parameters

`accountname` The name of the account.

Return value A `BVI_MRAccount` instantiation.

BVI_MRAccountManager::getAccountByName()

Get an account object using the account name.

JavaScript
```
BVI_MRAccount getAccountByName ( string accountname );
```

Java
```
public final BVI_MRAccount getAccountByName ( String accountname )
```

Java exception Throws BVWComponentException.

Parameters

`accountname` The name of the account.

Return value `BVI_MRAccount`.

BVI_MRAccount

The `BVI_MRAccount` component manages the data that characterizes a trading account. Designated individuals using the account can buy, sell, or do both in the market.

JavaScript
```
$BV1TO1/include/jsi/components/mraccount.jsi
```

Java `com.broadvision.components.BVI_MRAccount.*`

This interface has the following methods:

Method	Description	Page
addVisitorToAccount()	Add a member to an account.	87
removeVisitorFromAccount()	Remove the visitor from only the alternate account.	87

BVI_MRAccount::addVisitorToAccount()

Add the existing visitor to the current account with the specified preferences. The visitor must have been previously defined to the system in another account using `BV_VisitorMgr::newVisitor()`. This account is considered an alternate account for the visitor.

JavaScript

```
void addVisitorToAccount (
    string username,
    BVI_Properties preferences );
```

Java

```
public final void addVisitorToAccount (
    String username,
    BVI_Properties preferences )
```

Java exception Throws BVWComponentException.

Parameters

`username`	Specifies the name of the visitor to delete from the account object.
`preferences`	Specifies the user's preferences. Note that `MRPMT_PREF` cannot be specified at this time.

Return value None.

Remarks `MRPMT_PREF` cannot be specified in invoking `addVisitorToAccount()`. To set `MRPMT_PREF`, first add the visitor to the account, then invoke `allowUserToUsePaymentMethod()`, and finally invoke `setUserPreferences()`.

BVI_MRAccount::removeVisitorFromAccount()

Remove the visitor from the account only if the specified account is an alternate account for the visitor.

JavaScript

```
void removeVisitorFromAccount ( string username );
```

Java

```
public final void removeVisitorFromAccount ( string username )
```

Java exception Throws BVWComponentException.

Parameters

`username`	Specifies the name of the visitor to delete from the account object.

Return value None.

BVI_MROrgEntityMgr

The BVI_MROrgEntityMgr class creates and manages BVI_MROrgEntity components.

JavaScript `$BV1TO1/include/jsi/components/mrorgentitymgr.jsi`

Java `com.broadvision.components.BVI_MROrgEntityMgr.*`

This interface has the following methods:

Method	Description
getOrgEntityById()	Get the specified organizational entity object by its OID.
getRootEntity()	Get the root organizational entity object for the account.

BVI_MROrgEntityMgr::getOrgEntityById()

Gets the root organizational entity object for the account.

JavaScript
```
BVI_MROrgEntity getOrgEntityById ( long oid );
```

Java
```
BVI_MROrgEntity getOrgEntityById ( long oid )
```

Java exception Throws BVWComponentException.

Parameters

`oid`	The OID of the organizational entity object.

Return value A `BVI_MROrgEntity` object.

BVI_MROrgEntityMgr::getRootEntity()

Gets the root organizational entity object for the account.

JavaScript
```
BVI_MROrgEntity getRootEntity ();
```

Java
```
BVI_MROrgEntity getRootEntity ()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value A `BVI_MROrgEntity` object.

BVI_MROrgEntity

The BVI_MROrgEntity component describes organizational entities.

JavaScript `$BV1TO1/include/jsi/components/mrorgentity.jsi`

Java `com.broadvision.components.BVI_MROrgEntity.*`

This interface has the following methods:

Method	Description
addMember()	Add the visitor to the organizational entity members.
createSubordinate()	Create an organizational entity object subordinate to the current organizational entity object.
getSubordinates()	Get the list of subordinate organizational entities.
removeMember()	Remove the visitor from the organizational entity.

This interface has the following attributes:

Attribute	Description
FULL_PATHNAME	Contain the path name of the organizational entity within the account.
INTERNAL_CODE	Contain the internal code of the organizational entity.
NAME	Contain the name of the organizational entity.
OID	Contain the OID of the organizational entity.
SUPERIOR_OID	Contain the OID of the organizational entity superior to this one.

BVI_MROrgEntity::addMember()

Add the visitor to the organizational entity.

JavaScript `void addMember ( long userid );`

Java `void addMember ( long userid )`

Java exception Throws BVWComponentException.

Parameters

`userid`	The visitor's user ID; must belong to the account to which the organizational entity belongs.

Return value None.

BVI_MROrgEntity::createSubordinate()

Create a subordinate organizational entity object. The entity_type must be a valid type defined for this account.

JavaScript
```
BVI_MROrgEntity createSubordinate ( string name,
                                    string entity_type );
```

Java
```
BVI_MROrgEntity createSubordinate ( String name,
                                    String entity_type)
```

Java exception Throws BVWComponentException.

Parameters

`name`	The name; must be unique in its full pathname in the organization structure.
`entity_type`	The entity_type; must be a valid type defined for this account.

Return value `BVI_MROrgEntity`: A new child `BVI_MROrgEntity` object.

BVI_MROrgEntity::getSubordinates()

Get all of the direct subordinates for this organizational entity.

JavaScript
```
BVI_ValueList getSubordinates ();
```

Java
```
BVI_ValueList getSubordinates ()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value `BVI_ValueList`: contains the `BVI_MROrgEntity` objects representing the subordinate organizational entities.

BVI_MROrgEntity::removeMember()

Delete the visitor from the organizational entity.

JavaScript
```
void removeMember ( long userid );
```

Java
```
void removeMember ( long userid )
```

Java exception Throws BVWComponentException.

Parameters

`userid` the visitor's user ID.

Return value None.

BVI_MROrgEntity::FULL_PATHNAME

This attribute contains the full pathname of this organizational entity within the organization structure of the account (in '/x/y/z' format).

JavaScript
```
attribute string FULL_PATHNAME;
```

Java
```
String getFULL_PATHNAME; ()
void setFULL_PATHNAME ( String FULL_PATHNAME )
```

Java exception Throws BVWComponentException.

BVI_MROrgEntity::INTERNAL_CODE

This attribute contains the internal code of the organizational entity.

JavaScript
```
attribute string INTERNAL_CODE;
```

Java
```
String getINTERNAL_CODE ()
void setINTERNAL_CODE ( String INTERNAL_CODE )
```

Java exception Throws BVWComponentException.

BVI_MROrgEntity::NAME

This attribute contains the name of this organizational entity.

JavaScript
```
attribute string NAME;
```

Java
```
String getNAME ()
void setNAME ( String NAME )
```

Java exception Throws BVWComponentException.

BVI_MROrgEntity::OID

This attribute contains the OID of the organizational entity object.

JavaScript
```
readonly attribute long OID;
```

Java
```
long getOID ()
```

Java exception Throws BVWComponentException.

BVI_MROrgEntity::SUPERIOR_OID

This attribute contains the OID of the organizational entity superior to this organizational entity.

JavaScript
```
readonly attribute long SUPERIOR_OID;
```

Java
```
long getSUPERIOR_OID ()
```

Java exception Throws BVWComponentException.

4 Customizing the Visitor Configurable Home Page

InfoExchange Portal provides the following levels of customization and configuration for visitor configurable home pages:

- **JavaScript template or Java Server page**. A developer creates a home page `.jsp` template file by defining the page layout, including placing images, logos, and other static items into the `.jsp` file.
- **Site administration**. A site administrator configures a home page by choosing a `.jsp` template file for a specific type of visitor and creating groups of topics for those visitors.
- **Visitor customization**. A visitor uses a profile editor to choose which groups of topics to place in the home page, the position of each group on the page, and the order of the topics in each group.

This chapter explains the developer's tasks and the JavaScript and Java interfaces that can be used to customize a home page at the JavaScript or Java level. It contains the following topics:

See the *InfoExchange Portal Administrator's Guide* for information on administering home pages using the InfoExchange Portal Admin Tool.

Home page concepts

There are three major configurable parts in a home page:

- **Page Type**. A *page type* determines which script is used to generate the home page for a visitor.
- **Block**. A *block* is a logical group of topics that can be placed on a home page.
- **Topic**. A *topic* is a program.

Home pages also support *transient guests*, or visitors who are not members of the site.

See the *One-To-One Overview* for more information on transient guests.

Granting transient guests access to your site

Many sites allow a visitor to have limited access to a site without logging in. The BVAdvenTech sample application start page gives limited access to a transient guest, enables the transient guest to create an account with the site, and enables a registered member to log into the site through the `bvadventech.html` start page.

While user profiles for registered members are maintained in the database, profiles for transient guests are maintained in memory only. Accordingly, when a transient guest leaves the site, all known profile information about the visitor is discarded. This classification is used by sites where the visitor's identity is not integral to the application, such as sites that allow anonymous browsing.

Configuring InfoExchange Portal for transient guests

Each visitor's user account, including the transient guest user account, is associated with a user template. The user template defines the access groups, and qualifiers assigned to a new user account. Each user template is associated with a page type. The page type is where an administrator defines the information visitors see on their home pages.

The following must be done before a transient guest can "log into" your site:

- A user template must be assigned to the transient guest.

The *InfoExchange Administrator's Guide* contains information on setting qualifiers, as well as setting and configuring access groups.

When you load the InfoExchange Portal sample data you load additional data to support transient guests, including a "`Guest`" user template and a "`Guest`" page type. The user templates and page types for transient and registered visitors are administered using the InfoExchange Portal Admin Tool. The user template for transient guests is hard-coded in `login_adventech.jsp` and `guestHomePage.jsp`.

Before registering or logging in, a transient guest can:

- select the optional and default blocks
- control the layout of the selected blocks
- change the heading and subheading colors

When a transient guest registers, any heading and subheading colors the visitor has set are preserved and override the default heading and subheading colors from the home page script. Each user template points to a page type that points to the home page script for that visitor. This home page script is where you define the default heading and subheading colors. The user template does not define default heading and subheading colors.

Implementing transient guests in JavaScript

The BVAdvenTech sample application implements logging in of transient guests as follows:

1. The page *doc-root*`/bvadventech.html` contains code that redirects the visitor to the script `guestHomePage.jsp` (all on one line):

   ```
   <META http-equiv="refresh" content="0;
   URL=/cgi-bin/inetcgi/adventech/scripts/home/guestHomePage.jsp">
   ```

2. The script *script-root*`/adventech/scripts/home/guestHomePage.jsp` contains the logic that logs transient guests into the site using the user template.

   The `default_user_template` parameter in `$BV1TO1_VAR/etc/bv1to1.conf` sets the default user template to assign to transient guests in the service. The following code can be used to retrieve this value:

   ```
   var userTemplate;
   var cfg = new BVI_SystemConfiguration("Adventech");
   if (!Error.set) {
       var val = cfg.parameter("default_user_template", null);
       if (!Error.set) {
           userTemplate = val.stringValue;
       }
   }
   ```

   If there is an error in the call, an error message appears; otherwise, the process continues.

   ```
   tmpEpHomePage.epInitializeUser("Guest");
       if (Error.set) {
           createNewUserError = Error.reason;
           // If createNewUserError != "" then a problem exists with
           // the setup.
   }
   ```

3. The transient guest is created using the following code:

   ```
   var serviceName = "Adventech"

   var visitor = currentVisitor();
   var userId  = visitor.ID;

   setCurrentVisitor(visitor);
   setCurrentService(serviceName);
   ```

4. The following variables are used to display the home page:

```
// <script_root>/adventech/scripts/common/topNav.jsp
// defines the following variables that are used in
// this page, admin.jsp, and embedded scripts.
//
// (Objects)
// epHomePage   BVI_EPHomePage
// km_progMgr   BVI_KMProgramManager
// cmcMgr       BVI_CPAccessManager
// (User's data)
// serviceId    long    Service Id
// userId       long    User Id
// userName     string User's name
// head         string User's Heading color
// subheading   string User's SubHeading Color
// epHomePageScript string  Page Type's Home Page Script
```

5. If the log in is successful, the system refreshes to the visitor's home page:

```
<META http-equiv="refresh" content="0;
URL=/cgi-bin/inetcgi/adventech/jsp/home/homeBusinessType.jsp">
```

Implementing transient guests in Java

The BVAdvenTech sample application implements logging in of transient guests as follows:

1. The page *doc-root*`/bvadventech.html` contains code that redirects the visitor to the script `guestHomePage.jsp` (all on one line):

```
<META http-equiv="refresh" content="0;
URL=/cgi-bin/inetcgi/adventech/jsp/home/guestHomePage.jsp">
```

2. The script contains the logic that logs transient guests into the site using the "`Guest`" user template.

   The method `IEPUtils.epGetTransientGuestUserTemplate()` can be used to retrieve the service-specific parameter `default_user_template` from the file `$BV1TO1_VAR/etc/bv1to1.conf`. In the sample application the user template is set to "`Guest`."

```
String guestUserTemplate =
IEPUtils.epGetTransientGuestUserTemplate(guestServiceName);
```

   An error message appears if there is an error in the call; otherwise, the process continues.

   The method `IEPUtils.epGetTransientGuestUserTemplate()` is part of the Java classes that reside in the `com.broadvision.ieportal.util` Java package.

3. The transient guest is created using the following line of code:

```
BVI_VisitorManager vMgr = new BVI_VisitorManager();
BVI_Visitor guestVisitor = vMgr.newGuestVisitor(false);
long guestUserId = guestVisitor.getID;
```

The `BVI_VisitorManager` component is discussed in the *Developer's Guide to Components and Scripts*.

4. The Java Server Page `guestHomePage.jsp` assigns the transient guest a user template ID, access groups, initializes the visitor's home page blocks, and determines whether the new member can use the closed-loop process management feature.

```
BVI_EPHomePage epHomePage = new BVI_EPHomePage(guestServiceId,
guestUserId);
epHomePage.epinitializeUser("Guest");
```

If the visitor has customized these settings while setting up the page, the visitor's settings override the default settings.

Registering a new member

There is only a JavaScript version of the register user page.

Visitors to the BVAdvenTech site are automatically logged in as transient guests and are assigned the "`Guest`" user template. The transient guest's home page data is read from memory. Transient users are assigned negative user IDs, members are assigned positive IDs. For example:

Visitor/Password	User ID
miles/miles	261
transient guest	-110

The script *`script-root`*`/adventech/scripts/common/registernewuser.jsp` prompts the visitor for a user name and password, then registers them as a new user, converting them from a transient guest to a registered member. Finally, the script logs the newly-registered member into the site.

When a transient guest registers, only the heading and subheading colors he or she has set are kept. Any other changes made to the home page are replaced by those from the new user template when the new user account is created.

1. Once the transient guest has entered a user name and password, the JavaScript page creates the registered user account:

```
visitor.registerGuest(username, password, true);
```

2. The script then:
   - Assigns a user template ID to the visitor.
   - Assigns Publishing Center access groups.
   - Assigns default home page blocks.
   - Assigns qualifiers to user. Transient users share the same qualifier information.
   - Sets the visitor to be active (BV_KM_UPROF.KM_USER_DELETED=0).

```
var tmpEpHomePage = new BVI_EPHomePage(Session.serviceID, userId);
tmpEpHomePage.epInitializeUser("Visitor");
```

3. The script assigns a user template to the visitor's new account and flags it as "active":

```
prop.NAME  = name;
prop.EMAIL = email;
prop.EP_HEADING = Session.epHeading;
prop.EP_SUBHEADING = Session.epSubHeading;
visitor.updateProperties(prop);
```

4. The script retrieves the visitor's page type ID and home page script.

```
var prop = epHomePage.getUserPageTypeSettings();
if (Error.set) {
    error=Error.reason;
} else {
    epHomePageScript = prop.get("DISPLAY_SCRIPT");
    accepted=1;
}
```

Functions and methods for setting up transient guests

Use one of the following functions and methods to set up a transient guest to use your site:

- The JavaScript helper function bv_erm_get_transient_guest_user_template(), defined in the file *script-root*`/adventech/scripts/common/guestloginutils.jsp`.
- The function IEPUtils.epGetTransientGuestUserTemplate(), which is part of the Java classes in the `com.broadvision.ieportal.util` Java package.

The value of the transient guest user template is hard-coded to "`Guest`" when using the Java method.

bv_erm_get_transient_guest_user_template()

Gets the user template for the transient guest in the specified service.

JavaScript `bv_erm_get_transient_guest_user_template(serviceName)`

Parameters

`serviceName`	service name

Return value The user template name for the transient guest in the specified service.

Example `var userTemplate = bv_erm_get_transient_guest_user_template(serviceName);`

IEPUtils.epGetTransientGuestUserTemplate()

Gets the user template for the transient guest in the specified service.

```
String epGetTransientGuestUserTemplate(serviceName)
```

Parameters

`serviceName`	service name

Return value: The user template name for the transient guest in the specified service.

Example:

```
String guestUserTemplate =
IEPUtils.epGetTransientGuestTemplate(guestServiceName);
```

BVI_KMUserType

The BVI_KMUserType class is the underlying data type that contains user templates.

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; the Java classes reside in the `com.broadvision.ieportal.components` Java packages.

This component does not allow dynamic properties.

BVI_KMUserType::getAccessGroups()

Gets the access groups for the user template.

JavaScript:

```
BVI_ValueList getAccessGroups (string serviceId);
```

Java:

```
public final BVI_ValueList getAccessGroups (String serviceId);
```

Java exception: Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID of the user template

Return value: A BVI_ValueList object or null if an error occurs. Each element of the list is an OID of a Publishing Center access group.

Example:

```
var kmMgr = new BVI_KMAdminManager;
var userType = kmMgr.getUserType(serviceId, userTypeId);
var accessGroups = userType.getAccessGroups(serviceId);
```

BVI_EPHomePageManager

The BVI_EPHomePageManager class implements the methods for administering page types and blocks for home pages. Except for switchUserTemplate, these methods are reserved for BroadVision use.

This component does not allow dynamic properties.

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; Java classes reside in the `com.broadvision.ieportal.components` Java packages.

BVI_EPHomePageManager::switchUserTemplate()

Assigns the visitor the default blocks associated with the specified user template.

JavaScript

```
void switchUserTemplate(long userId, long userTypeId);
```

Java

```
public final void switchUserTemplate(long userId, long usertypeId)
```

Java exception

Throws BVWComponentException.

Parameters

`userId`	An integer containing the user ID
`userTypeId`	An integer containing the user template ID

Return value

None; on error, the error object is set.

Example

```
var EPHomePageManager = new BVI_EPHomePageManager(serviceId, userId);
// Assign the user template's default blocks to user.
```

BVI_EPHomePage

The BVI_EPHomePage class is used to add, update, and remove information from visitors' home pages. Visitors can update their block and topic information. This class is used by the home page scripts (*`script_root`*`/adventech/scripts/home/*.jsp` for JavaScript or *`script_root`*`/adventech/jsp/home/*.jsp` for Java Server pages) to display information to the visitor.

This component does not allow dynamic properties.

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; Java classes reside in the `com.broadvision.ieportal.components` Java package.

The following table lists the BVI_EPHomePage methods.

Constructors and destructor methods	
creator()	Creates a new BVI_EPHomePage object.
creator()	Returns a copy of an existing BVI_EPHomePage object.
clone()	Copies the specified BVI_EPHomePage object.
Block methods	
deleteBlock()	Deletes the visitor's Optional or Default block specified by the block OID.
getAdminBlocks()	Returns all of the Required (admin) block topics for the specified page type.
getNonAdminBlocks()	Returns all of the Optional and Default (non-admin) block topics for the specified page type.
getUserBlockLayout()	Returns a list of the Optional and Default blocks for a given page type.
getTypedBlockList()	Returns all blocks of a given type for a page type.
getUserBlockList()	Returns a list of the visitor's selected blocks for a given page type.
setAdminBlockExpandFlag()	Expands or collapses a Required (admin) block.
setBlockExpandFlag()	Expands or collapses an OPTIONAL or DEFAULT block. 0 - OPTIONAL 1 - DEFAULT
updateBlocks()	Updates a visitor's selection of Optional and Default blocks for a specific page type.
epInitializeUser()	Initializes user account settings.
Inbox methods	
inBoxMessages()	Retrieves all messages in the alert inbox for the specified user and alert type.
Page type methods	
getPageType()	Returns a page type's properties.
getUserPageTypeSettings()	Gets page type information for a given visitor.
Topic methods	
getSiteSelection()	Returns a list of site topics for a given block. Filters out the visitor's selected topics. Each element of the list is a BVI_Properties object.
getUserSelection()	Returns a list of user selected topics for a given block. Each element of the list is a BVI_Properties object.
updateTopics()	Updates a visitor's selection of topics for a specific block.

Required blocks are sometimes called Admin blocks in the BVI_EPHomePage methods. Optional and Default blocks are sometimes called non-admin blocks in the BVI_EPHomePage methods.

BVI_EPHomePage::creator()

Creates a new BVI_EPHomePage object.

JavaScript

```
creator(
    long serviceID
    long userID);
```

Java

```
public BVI_EPHomePage(
    long serviceID,
    long userID)
```

Java exception: Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

`serviceID`	An integer containing the ID for the current service.
`userID`	An integer containing the user ID for whom the page is being created.

Return value: A BVI_EPHomePage object, or null if an error occurs.

Example: The following code segment creates a new instance of `BVI_EPHomePage`, assigns it to the variable `epHomePage`, and stores it in the `Session` object.

```
var epHomePage = new BVI_EPHomePage(Session.serviceID, userId);
Session.epHomePage = epHomePage;
```

BVI_EPHomePage::creator()

Create a copy of a source BVI_EPHomePage object.

JavaScript

```
creator(BVI_EPHomePage ref);
```

Java

```
public BVI_EPHomePage(BVI_EPHomePage ref)
```

Java exception: Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

`ref`	The instance of BVI_EPHomePage to be copied. Must not be null.

Return value: A reference to a BVI_EPHomePage object, or null if an error occurs.

Example: The first line of this example creates a BVI_EPHomePage component called `epHomePage`. The second line uses the home page created in the first line.

```
var epHomePage = new BVI_EPHomePage(serviceID, userId);
var copyEPHomePage = new BVI_EPHomePage(epHomePage);
```

BVI_EPHomePage::clone()

Create a copy of the specified BVI_EPHomePage object.

JavaScript
```
BVI_EPHomePage clone();
```

Java
```
public final BVI_EPHomePage clone_J()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value A copy of the specified BVI_EPHomePage object, or null if an error occurs.

Example
```
var epHomePage = new BVI_EPHomePage(serviceId, userId);
var clone = epHomePage.clone();
```

BVI_EPHomePage::deleteBlock()

Deletes the visitor's Optional or Default block specified by the block OID.

JavaScript
```
void deleteBlock(long blockID);
```

Java
```
public final void deleteBlock(long blockOID)
```

Java exception Throws BVWComponentException.

Parameters

`blockOID`	An integer containing the block ID of the block to be deleted from the visitor's settings.

Return value None.

Example The following code segment deletes a block with an ID that has been assigned to the variable `tmpOID` after being marked for deletion by the visitor. This code segment assumes that an instance of BVI_EPHomePage has been initialized and assigned to the variable `epHomePage`.

 The block is removed only from the current visitor's home page.

```
var tmpOID;
if((tmpOID = Request.value('DELETE_OID')) != null) {
    epHomePage.deleteBlock(tmpOID);
    if( Error.set ) {
        error = Error.reason;
    }
}
```

BVI_EPHomePage::getAdminBlocks()

Returns all of the Required (admin) block topics for the specified `pageTypeID`.

JavaScript

```
BVI_ValueList getAdminBlocks(long pageTypeID);
```

Java

```
public final BVI_ValueList getAdminBlocks(long pageTypeID)
```

Java exception

Throws BVWComponentException.

Parameters

`pageTypeID`	An integer containing the page type ID for which the list of Required blocks is being requested.

Return value

A BVI_ValueList containing a list of BVI_Properties. (Returns null if an error occurs.) The BVI_ValueList represents a list of Required (Admin) block topics. Each BVI_Properties represents a topic inside a Required (Admin) block. The BVI_Properties objects contain the following elements:

BLOCK_NAME	Block name
BLOCK_ID	Block OID
PROGRAM_NAME	Program name
PROGRAM_ID	Program OID
EXPAND	Expand flag, which specifies whether the block is to be expanded or not. Valid values for this flag are: 0—Collapse (close) the block 1—Expand the block
NUM_OF_ENTRIES	Number of entries to display in the program
FRAGMENT_PATH	The block script path for displaying the program in the home page. This script can display images, stock quotes, tables, and so on if the administrator specifies the appropriate FRAGMENT_PATH
INTCOL1	Admin defined integer column
INTCOL2	Admin defined integer column
INTCOL3	Admin defined integer column
STRCOL1	Admin defined string column
STRCOL2	Admin defined string column
STRCOL3	Admin defined string column
PROGRAM_PAGE_PATH	The path to the script or URL for the program
PROGRAM_PAGE_TYPE	The type of page for the program. Valid values for this flag are: 0—Script 1—URL
TIMEOUT	Timeout at the program cache.
CACHEABLE	Cacheable for the program.

Visitors cannot select or deselect Required (Admin) blocks. Only the administrator can show or hide these blocks. However, a visitor can choose to collapse a required block on a personal home page.

Example

The following code segment retrieves the Required block topics for a visitor. This code segment assumes that an instance of BVI_EPHomePage has been initialized and assigned to the variable `epHomePage`.

```
// Retrieve visitor's Required Blocks
adminList = epHomePage.getAdminBlocks(pageTypeID);
if( Error.set ) {
    error = Error.reason;
}
```

BVI_EPHomePage::getNonAdminBlocks()

Returns all Optional and Default (non-Admin) blocks' topics for a given `pageTypeID`. These are the topics for the Optional and Default blocks.

JavaScript

```
BVI_ValueList getNonAdminBlocks(long pageTypeID);
```

Java

```
public final BVI_ValueList getNonAdminBlocks(long pageTypeID)
```

Java exception

Throws BVWComponentException.

Parameters

`pageTypeID`	An integer containing the page type ID for which the list of Optional and Default blocks is being requested.

Return value

A BVI_ValueList containing a list of BVI_ValueList, each element of the secondary (contained) value lists contains a BVI_Properties. (Returns null if an error occurs.) The first BVI_ValueList (which contains the other value lists) contains or points to a list of columns that appear on the visitor's home page. The secondary value lists contain the Optional or Default block topics for that list. Each BVI_Properties represents the data inside the Optional or Default (non-admin) block topics. The BVI_Properties objects contain the following elements:

BLOCK_NAME	Block name.
BLOCK_ID	Block OID.
PROGRAM_NAME	Program name.
PROGRAM_ID	Program OID.
EXPAND	Expand flag, which specifies whether the block is to be expanded or not. Valid values for this flag are: 0—Collapse (close) the block. 1—Expand the block.
NUM_OF_ENTRIES	Number of entries to display in the program.
FRAGMENT_PATH	The block script path for displaying the program in the home page. This script can display images, stock quotes, tables, and so on if the administrator specifies the appropriate FRAGMENT_PATH

INTCOL1	Admin defined integer column
INTCOL2	Admin defined integer column
INTCOL3	Admin defined integer column
STRCOL1	Admin defined string column
STRCOL2	Admin defined string column
STRCOL3	Admin defined string column
PROGRAM_PAGE_PATH	The path to the script or URL for the program
PROGRAM_PAGE_TYPE	The type of page for the program. Valid values for this flag are: 0—Script 1—URL
TIMEOUT	Timeout at the program cache.
CACHEABLE	Cacheable for the program.

Example

The following code segment retrieves the list of optional and default block topics for the visitor. This code segment assumes an instance of BVI_EPHomePage has been initialized and assigned to the variable epHomePage.

```
var list = epHomePage.getNonAdminBlocks(pageTypeID);
if( Error.set ) {
    error = Error.reason;
} else {
    var numberOfColumns = (list == null) ? 0 : list.length;
}

//loop through columns
for(II=0; II<numberOfColumns; II++) {
    list.cursor = II;
    var subList = list.objectValue;
    var subnColumns = (subList == null) ? 0 : subList.length;

    for(var JJ=0; JJ<subnColumns; JJ++) {
        // Each entry in subList list is a BVI_Properties component.
        // This entry represents a topic and has the following properties
        //   BLOCK_NAME             Block Name
        //   BLOCK_ID               Block OID
        //   PROGRAM_NAME           Program Name
        //   PROGRAM_ID             Program OID
        //   EXPAND                 Flag 0 - collapse 1- expand
        //   NUM_OF_ENTRIES         Number of Entries
        //   FRAGMENT_PATH          Embed Script Path
        //   INTCOL1-3              INTCOL1-3
        //   STRCOL1-3              STRCOL1-3
        //   PROGRAM_PAGE_PATH      A program's navigation path
        //   PROGRAM_PAGE_TYPE      A program page type

        subList.cursor = JJ;
        var prop       = subList.objectValue;

        var blockName       = prop.get("BLOCK_NAME");
        var blockID         = prop.get("BLOCK_ID").longValue;
        var programName     = prop.get("PROGRAM_NAME");
        var programID       = prop.get("PROGRAM_ID").longValue;
        var expand          = prop.get("EXPAND").longValue;
        var numberOfEntries = prop.get("NUM_OF_ENTRIES").longValue;
        var fragmentPath    = prop.get("FRAGMENT_PATH");
        var intcol1         = prop.get("INTCOL1").longValue;
        var intcol2         = prop.get("INTCOL2").longValue;
        var intcol3         = prop.get("INTCOL3").longValue;
        var strcol1         = prop.get("STRCOL1");
        var strcol2         = prop.get("STRCOL2");
        var strcol3         = prop.get("STRCOL3");
        var programPagePath = prop.get("PROGRAM_PAGE_PATH");
        var programPageType = prop.get("PROGRAM_PAGE_TYPE");
    }
}
```

BVI_EPHomePage::getUserBlockLayout()

Retrieves the visitor's Optional and Default blocks for a given page type.

JavaScript

```
BVI_ValueList getUserBlockLayout(long pageTypeID);
```

Java

```
public final BVI_ValueList getUserBlockLayout(long pageTypeID)
```

Java exception

Throws BVWComponentException.

Parameters

pageTypeID	An integer containing the ID for the page type for which the list of optional and default blocks is being requested.

Return value

A BVI_ValueList object, or null on error. Each element of this object is a BVI_Properties object with the following attributes:

BLOCK_NAME	The name of the block.
BLOCK_ID	Block OID.
EXPAND	Expand flag, which specifies whether the block is to be expanded or not. Valid values for this flag are: 0—Block is not expanded (closed) 1—Block is expanded (open).
COLUMN_ID	Column ID indicating which column to display the block in. The first column has an ID of 1.
INDEX	The element's index in the list starting at 0.

Example

The following example returns the visitor's Optional and Default blocks for the specified page type.

```
var selectionList = epHomePage.getUserBlockLayout(pageTypeID);
```

BVI_EPHomePage::getTypedBlockList()

Returns a list of all blocks of a given type for a given page type.

JavaScript

```
BVI_ValueList getTypedBlockList(
    long pageTypeID
    long blockType);
```

Java

```
public final BVI_ValueList getTypedBlockList(
    long pageTypeID,
    long blockType)
```

Java exception

Throws BVWComponentException.

Parameters

pageTypeID	An integer containing the page type ID for which the list of blocks is being requested.
blockType	An integer containing the block type. Valid values for this parameter are: 0—Optional. 1—Default. 2—Required.

Return value

A BVI_ValueList object, or null if an error occurs. Each element of the list is a BVI_Properties object with the following elements:

BLOCK_ID	Block OID
BLOCK_NAME	Block name

Example

The following example returns all Optional blocks for a pageType.

```
var list = epHomePage.getTypedBlockList( pageTypeID, 0 );
```

BVI_EPHomePage::getUserBlockList()

Returns a list of the visitor's selected blocks for a given page type. This method is used to determine if a user uses a selected block or not. This method is used on Optional or Default blocks.

JavaScript

```
BVI_ValueList getUserBlockList(
    long pageTypeID
    long blockType);
```

Java

```
public final BVI_ValueList getUserBlockList(
    long pageTypeID,
    long blockType)
```

Java exception

Throws BVWComponentException.

Parameters

pageTypeID	An integer containing the page type ID for which the list of blocks is being requested
blockType	An integer containing the block type for which the list of blocks is being requested. This method is used on Optional or Default blocks. If you pass in a block type representing a Required block, this method returns an error.

Return value

A `BVI_ValueList` object, or null if an error occurs. Each element of the list is a `BVI_Properties` object with the following elements:

BLOCK_ID	Block OID
COLUMN_ID	Column ID of the column containing the block. First column has ID of 1
EXPAND	Expand flag, which specifies whether the visitor wants the block expanded or not. Valid values for this flag are: 0—Block is not expanded (closed) 1—Block is expanded (open)
DISPLAY_ORDER	An integer representing the display order of the block within the column, starting at 0. A value of `n` indicates the visitor selected this block and its display order is `n`

Example

The following example returns all Optional blocks for a `pageType`.

```
var userList1 = epHomePage.getUserBlockList( pageTypeID, 0 );
```

BVI_EPHomePage::setAdminBlockExpandFlag()

Expands or collapses a Required block.

JavaScript

```
void setAdminBlockExpandFlag(
    long blockOID,
    long expandFlag);
```

Java

```
public final void setAdminBlockExpandFlag(
    long blockOID,
    long expandFlag)
```

Java exception

Throws BVWComponentException.

Parameters

`blockOID`	An integer containing the OID of the block to be expanded or collapsed.
`expandFlag`	Expand flag, which specifies whether the block is to be expanded or not. Valid values for this flag are: 0—Collapse (close) the block. 1—Expand the block.

Return value

None.

Example

The following code segment checks to see if the visitor has requested that a Required block be expanded or collapsed and uses setAdminBlockExpandFlag() to set the value of expandFlag to the value of ADMIN_ISEXPANDED for the specified block. This code segment assumes an instance of BVI_EPHomePage has been initialized and assigned to the variable epHomePage.

```
if((tmpOID = Request.value('ADMIN_EXPAND_OID')) != null) {
    var isExpanded = Request.value('ADMIN_ISEXPANDED');
    epHomePage.setAdminBlockExpandFlag(tmpOID, isExpanded);
}
```

BVI_EPHomePage::setBlockExpandFlag()

Expands or collapses an Optional or Default block.

```
void setBlockExpandFlag(
    long blockOID,
    long expandFlag);
```

Java

```
public final void setBlockExpandFlag(
    long blockOID,
    long expandFlag)
```

Java exception

Throws BVWComponentException.

Parameters

blockOID	An integer containing the OID of the block to be expanded or collapsed.
expandFlag	Expand flag, which specifies whether the block is to be expanded or not. Valid values for this flag are: 0—Collapse (close) the block. 1—Expand the block.

Return value

None.

Example

The following code segment checks to see if the visitor has requested that an Optional or Default block be expanded or collapsed and runs setBlockExpandFlag() to set the value of expandFlag to the value of ISEXPANDED for the specified block. This code segment assumes an instance of BVI_EPHomePage has been initialized and assigned to the variable epHomePage.

```
if((tmpOID = Request.value('EXPAND_OID')) != null) {
    var isExpanded = Request.value('ISEXPANDED');
    epHomePage.setBlockExpandFlag(tmpOID, isExpanded);
}
```

BVI_EPHomePage::updateBlocks()

Updates a visitor's selection of Optional or Default blocks for a specific page type.

JavaScript

```
void updateBlocks (
    long pageTypeID
    BVI_MultiValueTable values
);
```

Java

```
public final void updateBlocks(long pageTypeID, BVI_MultiValueTable
values)
```

Java exception

Throws BVWComponentException.

Parameters

pageTypeID	An integer containing the ID for the page type in which the blocks are being updated.
values	A BVI_MultiValueTable (BV_EP_UPROF_BLOCK) object that holds the new block values.

The attributes in the BV_EP_UPROF_BLOCK BVI_MultiValueTable are:

EP_BLOCK_ID	Block OID
EP_PAGETYPE_ID	Page Type ID for the page type containing the block
EP_COLUMN_ID	Column ID of the column containing the block
EP_DISPLAY_ORDER	An integer representing the display order. A value of n indicates the visitor selected this block and its display order is n.
EP_EXPAND	Expand flag, which specifies whether the block is to be expanded or not. Valid values for this flag are: 0—Block is not expanded (closed) 1—Block is expanded (open).

Return value

None

Example

In this example, BVI_EPHomePage.updateBlocks overwrites the visitor's existing entries in the BV_EP_UPROF_BLOCK table with new entries.

```
epHomePage.updateBlocks(pageTypeID, newTable);
if( Error.set ) {
    error = Error.reason;
}
```

BVI_EPHomePage::epInitializeUser()

Initializes user account settings.

JavaScript

```
void epInitializeUser(string userTemplateName);
```

Java
```
public final void epInitializeUser(String userTemplateName);
```

Java exception Throws BVWComponentException.

Parameters

`userTemplateName`	A string; the name of the user template.

Return value None

Remarks This method does the following:

- Converts the user template name into a user template ID.
- Assigns the user template ID to the visitor (BV_KM_UPROF.KM_USERTYPE_ID).
- Assigns the Publishing Center access groups associated with the user template to the visitor.
- Assigns the default blocks associated with the user template to the visitor.

If the visitor is not a transient guest, it assigns the qualifiers from the user template to the user account. Transient guests have negative user IDs; members have positive user IDs.

If the visitor is a registered member of the site, this sets BV_KM_UPROF.KM_USER_DELETED to 0 to allow the visitor to log into the site.

BVI_EPHomePage::inBoxMessages()

Retrieves all messages in the alert inbox for the specified visitor and alert type.

JavaScript
```
BVI_Table inBoxMessages(
    long userId,
    string alertType,
    long filter,
    long maxMessages);
```

Java
```
public final BVI_Table inBoxMessages(
    long userId,
    String alertType,
    long filter,
    long maxMessages)
```

Java exception Throws BVWComponentException.

Parameters

`userId`	The ID of the visitor for whom the messages are being retrieved.
`alertType`	The alert type for the messages to be retrieved. This method accepts only the following alert types: **bv_km_content_alert**—Filters out off-line and deleted content items. To get all content alerts, including alerts for off-line and deleted content, use `BVI_AlertManager::inBoxMessages`, which is defined in One-To-One Enterprise. **bv_km_category_alert**—Filters out deleted, off-line, and non-visible programs. A non-visible program is one where the navigation filters have been modified by an administrator so the visitor cannot see it.
`filter`	Determines which messages to retrieve. Valid values are: 0—Retrieve both old and new inbox messages. 1—Retrieve only new inbox messages.
`maxMessages`	Specifies the maximum number of messages to be retrieved. Specifying the value 0 results in all messages being retrieved.

Example

This example retrieves all content alert messages in the alert inbox for the specified user.

```
var inbox = epHomePage.inBoxMessages(userId, 'bv_km_content_alert',
    filter, maxCount);
```

BVI_EPHomePage::getPageType()

Returns a page type's properties.

JavaScript

```
BVI_Properties getPageType(long pageTypeID);
```

Java

```
public final BVI_Properties getPageType(long pageTypeID)
```

Java exception

Throws BVWComponentException.

Parameters

`pageTypeID`	An integer containing the ID of the page type for which the page type properties are being requested.

Return value

Returns a BVI_Properties object, or null if an error occurs. The BVI_Properties object has the following attributes.

PAGETYPE_ID	Page type OID.
PAGETYPE_NAME	Page type name.
DISPLAY_SCRIPT	Path of script for displaying home page.
NUM_OF_COLUMNS	Maximum number of columns in home page for page type.
NUM_OF_ENTRIES	Maximum number of entries to display for topics of required blocks.

Example

This methods retrieves a page type's properties. This example returns a `BVI_Properties` object with which you can retrieve the maximum number of columns this page type supports.

```
var newProps = epHomePage.getPageType(pageTypeID);
var maxNumberOfColumns = newProps.get("NUM_OF_COLUMNS").longValue;
```

BVI_EPHomePage::getUserPageTypeSettings()

Gets page type information for a given visitor.

JavaScript

```
BVI_Properties getUserPageTypeSettings();
```

Java

```
public final BVI_Properties getUserPageTypeSettings()
```

Java exception

Throws BVWComponentException.

Parameters

None.

Return value

Returns a BVI_Properties object, or null if an error occurs. The BVI_Properties object has the following attributes.

PAGETYPE_ID	Page type OID.
DISPLAY_SCRIPT	Path of script for displaying home page.

Example

This code segment from *`script-root`*`/adventech/scripts/login_adventech.jsp` runs getUserPageTypeSettings() and assigns the results to the variable `prop`. It is assumed an instance of BVI_EPHomePage has been initialized and assigned to the variable `epHomePage`.

```
var prop = epHomePage.getUserPageTypeSettings();
```

BVI_EPHomePage::getSiteSelection()

JavaScript

```
BVI_ValueList getSiteSelection(long blockID);
```

Returns a list of the topics not selected by the user for a given optional or default block.

When you combine the two lists from getSiteSelection and getUserSelection(), the result represents all available topics for a block. The getUserSelection() method returns the visitor-selected topics for a given block. The visitor's selected topics are stored in BV_EP_UPROF_CHPRG. The getSiteSelection method returns the user non-selected topics for a given block. If you call getSiteSelection with a Required block it returns an error.

Java

```
public final BVI_ValueList getSiteSelection(long blockID)
```

Java exception

Throws BVWComponentException.

Parameters

`blockID`	An integer containing the block ID for which the list of topics is being requested.

Return value A BVI_ValueList object, or null if an error occurs. Each element of the list is a BVI_Properties object with the following elements:

PROGRAM_ID	Program OID.
PROGRAM_NAME	Program Name.
NUM_OF_ENTRIES	Number of entries to display.
DISPLAY_ORDER	Display order.

Remarks This method excludes the visitor's selected topics.

Example
```
var siteList = epHomePage.getSiteSelection(blockOID)
```

BVI_EPHomePage::getUserSelection()

Returns a list of user selected topics for a given block. Only use the getUserSelection method on Optional or Default Blocks. Calling getUserSelection with a Required Block returns an error.

JavaScript
```
BVI_ValueList getUserSelection(long blockID);
```

Java
```
public final BVI_ValueList getUserSelection(long blockID)
```

Java exception Throws BVWComponentException.

Parameters

`blockID`	An integer containing the block ID for which the list of topics is being requested.

Return value A BVI_ValueList object, or null if an error occurs. Each element of the list is a BVI_Properties object with the following elements:

PROGRAM_ID	Program OID
PROGRAM_NAME	Program Name
NUM_OF_ENTRIES	Number of entries to display
DISPLAY_ORDER	Display order

This method includes the visitor's selected topics.

Example
```
var userList = epHomePage.getUserSelection( blockOID );
```

BVI_EPHomePage::updateTopics()

Updates a visitor's selection of topics for a specific page type.

JavaScript

```
void updateTopics (
    long pageTypeID
    BVI_MultiValueTable values);
```

Java

```
public final void updateTopics(
    long pageTypeID,
    BVI_MultiValueTable values)
```

Java exception

Throws BVWComponentException.

Parameters

pageTypeID	An integer containing the ID for the page type in which topics are being updated.
values	A BVI_MultiValueTable object that holds the new topic values.

Each row of the table is a BVI_MultiValueRow object with the following attributes (from BV_EP_UPROF_CHPRG):

EP_CHPRG_ID	Program OID.
EP_BLOCK_ID	Block OID.
EP_PAGETYPE_ID	Page Type ID.
EP_ENTRY_NUM	Number of entries in this topic
EP_DISPLAY_ORDER	An integer representing the display order of the topic within its block. A value of n indicates the visitor selected this topic and its display order is n.

Return value

None

Example

The following code segment updates the topics for the visitor's home page by:

1. Requesting the blocks the visitor submitted.

```
var rightIndexes = Request.value('rightIndexes'); //submitted blocks
// get blocks from current visitor's profile
var userBlocks = currentVisitor().BV_EP_UPROF_CHPRG;
var newTable = null;
if( userBlocks == null ) {
    error = "Error: error updating user profile.";
} else {
    var pschema = userBlocks.schema //get schema for selected blocks
    if (pschema == null) {
        error = "Error: error updating user profile.";
    } else {
        newTable = new BVI_MultiValueTable(pschema);
        if (newTable == null) {
            error = "Error: error updating user profile.";
        }
    }
}
```

2. Retrieving the current multivalue table, BV_EP_UPROF_CHPRG, in the line:
   `var userBlocks = currentVisitor().BV_EP_UPROF_CHPRG;`

3. Retrieving the BV_EP_UPROF_CHPRG table schema, in the line:
   `var pschema = userBlocks.schema`

4. Creating a new BV_EP_UPROF_CHPRG table, in the line:
   `newTable = new BVI_MultiValueTable(pschema);`

5. Constructing a new row for the table.

```
//tokenize the block information in rightIndexes
if( !isEmptyString(rightIndexes) && error == "" ) {
    var tmpList = Request.tokenize(rightIndexes, '#');

    for(i=0; i<tmpList.length(); i++) {
        var entry = tmpList.get(i);
        var subList = Request.tokenize(entry, ':');
        var oid = subList.get(0);
        var num = subList.get(1);
            var newRow = new BVI_MultiValueRow(pschema);
        if( newRow != null ) {
            newRow.EP_CHPRG_ID    = oid;
            newRow.EP_BLOCK_ID    = blockOID;
            newRow.EP_PAGETYPE_ID = pageTypeID;
            newRow.EP_ENTRY_NUM   = num;
            newRow.EP_DISPLAY_ORDER = i;

            newTable.append(newRow);
        }
    }
}

// BVI_EPHomePage.updateTopic removes all previous
// settings and inserts the new values.
epHomePage.updateTopics(blockOID, newTable);
```

6. Populating the row with data. The data are topics for Optional or Default blocks.

7. Appending the new row to the table, in the line:
   `newTable.append(newRow);`

8. Updating the visitor's settings, in the line:
   `epHomePage.updateTopics(blockOID, newTable);`

BVI_EPTextCache

Whenever a visitor's home page is displayed, the embedded scripts of the programs selected on the home page fetch content and produce HTML. Because the home page is repeatedly visited, keeping frequently accessed content that is already displayed, cached in memory, or within the file system, can significantly improve the performance of a Web site.

InfoExchange Portal provides a system to support caching text data, such as HTML text produced by an embedding script, for each visitor. Embedded scripts in each program can use it to cache the already-generated HTML for each visitor. Whenever the content of a program on a visitor's home page is cached, the embedded script can quickly pull out the HTML text from the cache and pass it to the home page. Using a text cache, computing and displaying a home page involves retrieving the already-generated HTML content items of a program from the text cache, assembling them, and displaying them. However, if the content of a program keeps changing, the embedded script should not cache the HTML text. Instead, it has to produce new content each time the home page is displayed.

The BVI_EPTextCache class is used to administer, write to, and access data from the text cache. This class is used by the home page scripts (*script_root*/adventech/scripts/home/*.jsp for JavaScript and *script_root*/adventech/jsp/home/*.jsp for Java Server Pages) to cache end-user information for repeated display.

This component does not allow dynamic properties.

The JavaScript component interface files are located in $BV1TO1/include/jsi/bv/webapps; Java classes reside in the com.broadvision.ieportal.components Java package.

The following table lists the BVI_EPTextCache methods.

Method	Functionality
creator()	Creates a BVI_EPTextCache object for the default text cache.
creator()	Create a BVI_EPTextCache object for a named text cache.
write()	Stores text data in the text cache.
read()	Retrieves text data from the text cache.
setSize()	Sets text cache memory size limit in all running Interaction Managers.
getSize()	Gets text cache memory size limit in all running Interaction Managers.
setStatus()	Sets text cache status in all running Interaction Managers.
getStatus()	Gets text cache status for all running Interaction Managers.
setCacheLevel()	Sets the text cache level in all running Interaction Managers.
getCacheLevel()	Gets the text cache level for all running Interaction Managers.
flush()	Flushes an object from the text cache in all running Interaction Managers.
flushCache()	Flushes the entire text cache in all running Interaction Managers.

BVI_EPTextCache::creator()

Creates a BVI_EPTextCache object for the default text cache.

JavaScript

```
creator()
```

Java

```
public BVI_EPTextCache()
```

Java exception

Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

None.

Return value

A BVI_EPTextCache object, or null if an error occurs.

Example

The following code segment creates a new instance of BVI_EPTextCache.

```
var tc = new BVI_EPTextCache();
```

BVI_EPTextCache::creator()

Create a BVI_EPTextCache object for the named text cache.

JavaScript

```
creator(String cacheName);
```

Java

```
public BVI_EPHomePage(String cacheName)
```

Java exception

Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

`cacheName`	The instance of BVI_EPTextCache to be copied. Must not be null.

Return value

A BVI_EPTextCache object, or null if an error occurs.

Example

The first line of this example creates a BVI_EPTextCache component called `tc`. The second line uses the text cache created in the first line.

```
var tc = new BVI_EPTextCache(cacheName);
var tc0 = new BVI_EPTextCache();
```

BVI_EPTextCache::write()

Stores text data in the text cache.

JavaScript

```
void write(string key, string text);
```

Java

```
public final void write(String key, String text);
```

Java exception

Throws BVWComponentException.

Parameters

`key`	A unique string identifying the text data to be cached. It is required to be unique in the cache.
`text`	The text data to be cached.

Return value

None.

BVI_EPTextCache::read()

Retrieves text data from the text cache.

JavaScript

```
string read(
    string key,
    long timeOut,
    string formString,
    string linkString);
```

Java

```
public final String read(
    String key,
    long timeOut,
    String formstring,
    String linkString);
```

Java exception

Throws BVWComponentException.

Parameters

`key`	A unique string identifying the text string to be cached.
`timeOut`	Number of seconds specifying when the cached text data expires. 0 means to return the cached text data without checking the timestamp. Otherwise, it only returns the cached text data when it was cached within 'timeOut' seconds.
`formString`	A special string containing the session ID and engine ID. You can obtain this string by using Session.Location.formString(). This special string is used with an HTML form on a page in a One-To-One Enterprise site.
`linkString`	A special string containing session ID and engine ID. You can obtain this string by using Session.Location.linkString(). This special string is used in hyperlinks pointing to other locations within a One-To-One Enterprise site.

Return value

When `formString` has a non-null value, this method replaces `formString` with the specified `formString` in the returned cached text data. When `linkString` has a non-null value, this method replaces `linkString` with the specified `linkString` in the returned cached text data. If the text data identified with `key` does not exist in the text cache or was cached `timeOut` seconds ago, this method returns null.

Remarks

A `formString` has the following format:

```
<input type=hidden name="BV_SessionID" value="...">
<input type=hidden name="BV_EngineID" value="...">
```

A `linkString` has the following format:

```
BV_SessionID=@@@@.....@@@@&BV_EngineID=....
```

BVI_EPTextCache::setSize()

Sets text cache memory size limit in all running Interaction Managers.

JavaScript
```
void setSize( long size );
```

Java
```
public final void setSize( long size );
```

Java exception Throws BVWComponentException.

Parameters

size	An integer representing the size of the text cache memory. It is the number of text data allowed to stay in memory of each Interaction Manager.

Return value None.

BVI_EPTextCache::getSize()

Gets text cache memory size limit.

JavaScript
```
long getSize();
```

Java
```
public final long getSize();
```

Java exception Throws BVWComponentException.

Parameters None.

Return value An integer representing the size of the text cache memory. This is the number of text data allowed to stay in cache memory.

BVI_EPTextCache::setStatus()

Sets text cache status in all running Interaction Managers.

JavaScript
```
void setStatus( long status );
```

Java
```
public final void setStatus( long status );
```

Java exception Throws BVWComponentException.

Parameters

`status`	An integer representing the text cache status. Valid values are: 0 - disabled 1 - enabled

Return value None.

BVI_EPTextCache::getStatus()

Gets text cache status.

JavaScript
```
long getStatus()
```

Java
```
public final long getStatus()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value An integer representing the text cache status. Valid values are:

0 - disabled

1 - enabled

BVI_EPTextCache::setCacheLevel()

Sets the text cache level.

JavaScript
```
void setCacheLevel( long cacheLevel );
```

Java
```
public final void setCacheLevel( long cacheLevel );
```

Java exception Throws BVWComponentException.

Parameters

`cacheLevel`	An integer representing the text cache level. Valid values are: 0 - memory only 1 - memory and file 2 - file only

Return value None.

BVI_EPTextCache::getCacheLevel()

Gets the text cache level for all running Interaction Managers.

JavaScript
```
long getCacheLevel();
```

Java
```
public final long getCacheLevel();
```

Java exception Throws BVWComponentException.

Parameters None.

Return value An integer representing the text cache level. Valid values are:

0 - memory only
1 - memory and file
2 - file only

BVI_EPTextCache::flush()

Flushes an object from the text cache in all running Interaction Managers.

JavaScript
```
void flush( string key );
```

Java
```
public final void flush( String key );
```

Java exception Throws BVWComponentException.

Parameters

`key`	A unique string identifying the text data to be flushed.

Return value None.

BVI_EPTextCache::flushCache()

Flushes the entire text cache in all running Interaction Managers.

JavaScript
```
void flushCache();
```

Java
```
public final void flushCache();
```

Java exception Throws BVWComponentException.

Parameters None.

Return value None.

5 Customizing the Project Collaboration Page

This chapter explains how to customize the BroadVision InfoExchange Portal project collaboration page. Topics covered include the files associated with the project collaboration page, the application programmer interface (API), and the methods associated with these files.

The following topics are covered in this chapter:

Note that this is not a fully featured Project Management application, but a project collaboration feature.

Introduction

An employee, such as a Project Manager, can create a Project Page to share information about a project with other employees, customers, or partners. Shared information about a project can include such things as tasks, meetings, announcements, programs, products and discussions. A project collaboration page pertains to a single project with a life cycle containing one or more phases. More than one phase can be in progress at one time.

A phase in a project collaboration page typically contains these types of data:

- time ordered lists of tasks which have an associated title, description, due date, and status (such as complete)
- time ordered lists of meetings, where each has a title, description, meeting date, agenda and meeting minutes section
- time ordered lists of announcements, sorted by creation date, which have an associated title and description
- a list of programs

In addition, the project collaboration page contains:

- a list of project groups, to which participants can belong
- a list of participants, where only participants can view the contents in a project

- contact lists with links to people who are contacts for a project, where the list is a subset of all the participants of a collaboration
- list of owners; an owner is a participant with special privileges, such as the ability to add and delete phases
- list of discussion groups
- list of products which may involve the project
- a list of programs associated with one or more phases of the project

BVI_EPProjectManager

The BVI_EPProjectManager object enables you to manipulate various aspects of a project collaboration page based on the ID or name of the project.

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; Java classes reside in the `com.broadvision.ieportal.components` Java packages.

```
implementation
{
    include <jsrt/properties_i.hh>
    include <jsrt/valuelist_i.hh>
    include "epprojectmanager_i.hh"
}

implementation { implementation_type = BVC_EPProjectManager; }
```

Constructor methods	
creator()	Creates a new BVI_EPProjectManager object.
creator()	Creates a copy of the specified BVI_EPProjectManager object.
clone()	Creates a copy of this BVI_EPProjectManager object.
Project methods	
addProject()	Adds a new project and returns the new project ID for the new project.
updateProject()	Updates a project.
deleteEntireProject()	Deletes an entire project, including its phases (including tasks, meetings, announcements).
getProject()	Gets a project by its name or ID.
findUserProject()	Obtains a list of projects that include the user as a participant.
Phase methods	
addPhase()	Adds a phase to a project.
updatePhase()	Updates a phase of a project.
deleteEntirePhase()	Deletes a phase, including its tasks, meetings, and announcements.
getPhase()	Gets a phase object by phase ID.
getPhaseInProject()	Gets all the phases related to a project.
Participant methods	
addParticipant()	Add a participant to a project.
updateParticipant()	Update the status of a project participant.

deleteParticipant()	Remove a participant from a project.
isParticipant()	Verifies a user ID is a participant in a project.
getParticipantInProject()	Get all the participants for a project.
getContactInProject()	Get the contact persons for a project.
Project group methods	
getGroupInProject()	Gets the groups for a specific project.
getGroupForUser()	Gets the groups to which the visitor belongs.
deleteGroup()	Deletes the specified group.
Task methods	
getTaskInPhase()	Returns the tasks related to a phase in a project.
getTaskInProject()	Returns a list of all tasks related to a project.
getAssignedTaskInPhase()	Returns the assigned tasks related to a phase in the project.
getAssignedTaskInProject()	Returns a list of all the assigned tasks related to the project.
Meeting methods	
getMeetingInPhase()	Finds all the meetings related to a phase.
getMeetingInProject()	Finds all the meetings related to a project.
Announcement methods	
getAnnouncementInPhase()	Finds all the announcements related to a phase.
getAnnouncementInProject()	Finds all the announcements related to a project.
Discussion methods	
getDgroupInProject()	Finds all the discussion groups the visitor may see in the project.
Miscellaneous methods	
isOwnerofProject()	Verifies whether the user ID is an owner of the project.
flushContent()	Flush the caches of all running Interaction Managers for a content item.

Using constructor methods

The BVI_EPProjectManager constructor methods enable you to create or duplicate a BVI_EPProjectManager object.

Do not call a constructor method directly. Instead, to call a constructor, use `new method` where *method* is the name of the constructor method.

Constructor methods	Functionality
creator()	Creates a new BVI_EPProjectManager object.
creator()	Creates a copy of the specified BVI_EPProjectManager object.
clone()	Creates a copy of this BVI_EPProjectManager object.

BVI_EPProjectManager::creator()

Creates a new BVI_EPProjectManager object.

JavaScript
```
BVI_EPProjectManager creator()
```

Java
```
public BVI_EPProjectManager creator()
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError.

Parameters None.

Return value A BVI_EPProjectManager object, or null if an error occurs.

BVI_EPProjectManager::creator()

Creates a copy of the specified BVI_EPProjectManager object.

JavaScript
```
BVI_EPProjectManager creator(BVI_EPProjectManager ref)
```

Java
```
public BVI_EPProjectManager creator(BVI_EPProjectManager ref)
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError.

Parameters

`ref`	An existing BVI_EPProjectManager object.

Return value A BVI_EPProjectManager object, or null if an error occurs.

BVI_EPProjectManager::clone()

Creates a copy of this BVI_EPProjectManager object.

JavaScript
```
BVI_EPProjectManager clone()
```

Java
```
public final BVI_EPProjectManager clone_J()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value A copy of the BVI_EPProjectManager object, or null if an error occurs.

Using project methods

The BVI_EPProjectManager project methods let you add, update, delete, or get a project.

Project methods	Functionality
addProject()	Adds a new project and returns the project ID for the new project.
updateProject()	Updates a project.
deleteEntireProject()	Deletes an entire project, including its phases (including tasks, meetings, announcements).
getProject()	Gets a project by its name or ID.
findUserProject()	Obtains a list of projects that include the user as a participant.

BVI_EPProjectManager::addProject()

Adds a new project and returns the project ID for the new project

JavaScript

```
long addProject(
    string serviceId,
    long userId,
    BVI_EPProject project );
```

Java

```
public final long addProject(String serviceId, long userId, BVI_EPProject
project)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	String; ID or name of the current service.
`userId`	Integer; the user ID number.
`project`	the definition of the project to be added to the service.

Return value

A integer containing the project ID for the added project.

Example

```
var user = currentVisitor;
var userId = user.ID;
var serviceId = Session.serviceID;
var projectMgr = Session.epProjectManager;
var rc;
var i;
var len;

// ---------------------------------------------------------------
// Retrieve the project information from the Session object
// and create a new project.
// ---------------------------------------------------------------
var project = new BVI_EPProject;
project.status = 1;
project.projectName = Session.ep_pw_projectName;
project.projectDescription = Session.ep_pw_projectDesc;
project.projectGoal = Session.ep_pw_projectGoal;
project.projectIconpath = Session.ep_pw_projectIconpath;

var projectId = projectMgr.addProject(serviceId, userId, project);
```

BVI_EPProjectManager::updateProject()

Updates a project.

JavaScript

```
void updateProject(string serviceId,
                   long userId,
                   BVI_EPProject project );
```

Java

```
public final void updateProject(String serviceId, long userId,
BVI_EPProject project)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	String; ID or name of the current service.
`userId`	Integer; the user ID number.
`project`	The information about a project in the service; this information replaces the current definition of the project.

Return value

None.

Example

```
var projectMgr = Session.epProjectManager;

if (currentPhaseId != null)
{
// ---------------------------------------------------------------
//update a project by changing which phase the project belongs to
// ---------------------------------------------------------------
    project = projectMgr.getProject(serviceName, projectId);

    if (Error.set) epErrorSystem;

    project.projectCurrentphase = currentPhaseId;
    projectMgr.updateProject(serviceName, userId, project);

    if (Error.set) epErrorSystem;

}
```

BVI_EPProjectManager::deleteEntireProject()

Deletes an entire project, including its phases (including groups, tasks, meetings, and announcements) and writes to the `$BV1TOl_VAR/logs/host/eplog` file the commands for deleting the project's tasks, meetings, and announcements attachment files, and to the `$BV1TOl_VAR/logs/host/dflag` file, writes the commands for deleting the project's discussion group attachment files. The administrator needs to manually delete the attachment files from the Web server.

JavaScript

```
void deleteEntireProject(
    string serviceId,
    long userId,
    string projectId );
```

Java

```
public final void deleteEntireProject(
    String serviceId,
    long userId,
    String projectId)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	String; ID or name of the current service.
`userId`	Integer; the user ID number.
`projectId`	String; ID or name of the project being deleted.

Return value

None.

Example

```
if (action == "delete")
    {
        var projectId = Request.value('projectId');
        if (projectId == null) epErrorMissingBrowserData( 'projectId' );
        projectMgr.deleteEntireProject(serviceName, userId, projectId);
        if (Error.set) epErrorSystem;
    }
```

BVI_EPProjectManager::getProject()

Gets a project by its name or ID.

JavaScript

```
BVI_EPProject getProject(string serviceId, string projectId);
```

Java

```
public final BVI_EPProject getProject(String serviceId, String projectId)
```

Java exception — Throws BVWComponentException.

Parameters

`serviceId`	String; ID or name of the current service.
`projectId`	String; ID or name of the project.

Return value — Returns a BVI_EPProject object.

Example

```
var project = projectMgr.getProject(serviceName, projectId);
```

BVI_EPProjectManager::findUserProject()

Obtains a list of projects for a user ID.

JavaScript

```
BVI_ValueList findUserProject(
    string serviceId,
    long userId);
```

Java

```
public final BVI_ValueList findUserProject(
    String serviceId,
    long userId);
```

Java exception — Throws BVWComponentException.

Parameters

`serviceId`	String; ID or name of the current service.
`userId`	Integer; the user ID number.

Return value — Returns a list of BVI_EPProject objects. The BVI_ValueList contains a BVI_EPProject object for each BVI_Value in the list. These are the projects that the user is a participant in.

Example

```
var serviceName = Session.serviceName;
var projectList = projectMgr.findUserProject(serviceName, user.ID);
```

Remarks

Use BVI_Value::objectValue to reference the BVI_EPProject object.

Using phase methods

The BVI_EPProjectManager phase methods let you add, update, delete, or get a phase of a project.

Phase methods	Functionality
addPhase()	Adds a phase to a project.
updatePhase()	Updates a phase of a project.
deleteEntirePhase()	Deletes a phase, including the tasks, announcements and meetings in that phase.
getPhase()	Gets a phase object by phase ID.
getPhaseInProject()	Gets all the phases of a project.

BVI_EPProjectManager::addPhase()

Adds a phase to a project.

JavaScript

```
long addPhase(
    string serviceId,
    long userId,
    BVI_EPPhase phase);
```

Java

```
public final long addPhase(
    String serviceId,
    long userId,
    BVI_EPPhase phase);
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	String; ID or name of the current service.
userId	Integer; the user ID number.
phase	BVI_EPPhase object defining the phase to be added to the project with the ID contained in phase.projectId.

Return value

Returns an integer containing the new phase ID for the new phase.

Example

```
if (Request.value("actionSave") != null)
{
    var projectMgr = Session.epProjectManager;

    //
    // Retrieve the user's entries from the form.
    //

    var phaseName = Request.value("phaseName");
    var phaseDesc = Request.value("phaseDesc");
    var phaseGoal = Request.value("phaseGoal");

    //
    // Initialize a new phase with the user's entries.
    //

    var phase = new BVI_EPPhase;

    phase.phaseName = stringTrim(phaseName);
    phase.phaseDescription = phaseDesc;
    phase.phaseGoal = phaseGoal;
    phase.projectId = projectId;
    phase.status = 1;   // 0="Off-line", 1="On-line"

    //
    // Add the new phase.
    //

    var phaseId = projectMgr.addPhase(Session.serviceName, userId, phase);

    if (Error.set)
    {
        if (Error.code == 17204) epErrorMessageId1(6810, phaseName);
        epErrorSystem;
    }

    //
    // After successfully adding a new phase, we display the list of phases.
    //

    visitScript('/adventech/scripts/project/phases.jsp');
}
```

BVI_EPProjectManager::updatePhase()

Updates a phase of a project.

JavaScript

```
void updatePhase(
    string serviceId,
    long userId,
    BVI_EPPhase phase);
```

Java

```
public final void updatePhase(
    String serviceId,
    long userId,
    BVI_EPPhase phase);
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	String; ID or name of the current service.
`userId`	Integer; the user ID number.
`phase`	BVI_EPPhase object to be updated.

Return value

None

Example

```
var phase = projectMgr.getPhase (serviceName, phaseId);
if (error.set) epErrorSystem;

if (Request.value("actionSave") != null)
{
    var phaseName = Request.value("phaseName");
    var phaseDesc = Request.value("phaseDesc");
    var phaseGoal = Request.value("phaseGoal");

    phase.phaseName = stringTrim(phaseName);
    phase.phaseDescription = phaseDesc;
    phase.phaseGoal = phaseGoal;

    //
    // Update the phase.
    //

    projectMgr.updatePhase(serviceName, userId, phase);

    if (Error.set)
    {
        if (Error.code == 17204) epErrorMessageId1(6810, phaseName);
        epErrorSystem;
    }

    //
    // After successfully updating a phase, we display the list of
phases.
    //

    visitScript('/adventech/scripts/project/phases.jsp');
}
```

BVI_EPProjectManager::deleteEntirePhase()

Deletes a phase, including the tasks and announcements in that phase.

JavaScript

```
void deleteEntirePhase(
    string serviceId,
    long userId,
    string phaseId );
```

Java

```
public final void deleteEntirePhase(
    String serviceId,
    long userId,
    string phaseId );
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	String; ID or name of the current service.
userId	Integer; the user ID number.
phaseId	String; ID of the phase to be deleted.

Return value

None.

Remarks

The script is responsible for checking that the userID is a project owner.

Example

```
if (action == 'delete')
{
    //
    // Get the ID of the phase to delete.
    //

    var phaseId = Request.value('phaseId');

    if (phaseId == null) epErrorMissingBrowserData( 'phaseId' );

    //
    // Delete the phase.
    //

    projectMgr.deleteEntirePhase(serviceName, userId, phaseId);

    if (Error.set) epErrorSystem;
}
```

BVI_EPProjectManager::getPhase()

Gets a phase object by phase ID.

JavaScript
```
BVI_EPPhase getPhase(string serviceId, string phaseId);
```

Java
```
public final BVI_EPPhase getPhase(String serviceId, String phaseId);
```

Java exception Throws BVWComponentException.

Parameters

serviceId	String; ID or name of the current service.
phaseId	String; ID or name of the phase object to be retrieved.

Return value A phase object.

Example
```
var phase = projectMgr.getPhase(serviceName, phaseId);
```

BVI_EPProjectManager::getPhaseInProject()

Gets all the phases related to a project.

JavaScript
```
BVI_ValueList getPhaseInProject(
    string serviceId,
    string projectId);
```

Java
```
public final BVI_ValueList getPhaseInProject(
    string serviceId,
    string projectId);
```

Java exception Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
projectId	String: ID or name of the project.

Return value A BVI_ValueList containing a BVI_EPPhase object for each BVI_Value in the list. These are the phases of the project.

Example
```
var phaseList = projectMgr.getPhaseInProject(serviceName, projectId);
```

Using participant methods

The BVI_EPProjectManager participant methods let you add, update, delete, verify, or get participants or contact persons related to a project or its collaboration page.

Participant methods	Functionality
addParticipant()	Add a participant to a project.
updateParticipant()	Update the status of a project participant.
deleteParticipant()	Remove a participant from a project.
isParticipant()	Verifies a user ID is a participant in a project.
getParticipantInProject()	Gets all the participants for a project.
getContactInProject()	Gets the contact persons related to a project collaboration page.
getGroupForUser()	Gets the groups to which the visitor belongs.
deleteGroup()	Deletes the specified group.

BVI_EPProjectManager::addParticipant()

Adds a participant to a project.

JavaScript

```
long addParticipant(
    string serviceId,
    long userId,
    string projectId,
    boolean contact,
    boolean owner,
    string contactInfor);
```

Java

```
public final long addParticipant(
    String serviceId,
    long userId,
    String projectId,
    boolean contact,
    boolean owner,
    String contactInfor);
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
userId	Integer; user account ID.
projectId	String; ID of the project.

`contact`	Boolean; true means the participant being added is a contact person for the project.
`owner`	Boolean; true means the participant being added is an owner of the project.
`contactInfor`	String; includes information about the participant, such as the participant's title.

Return Value

A status code indicating whether the participant was added to the project. A value of "0" means success; a value of "1" means failure.

Example

```
// --------------------------
// parse a list of users and add them
// as participants of the project
// --------------------------
for (i = 0; i < numberSelected; i++)
{
    var tokenList = Request.tokenize(userList.get(i), '%');
    var memberId = parseInt(tokenList.get(1));

    //
    // Check if the user is already  with the project.
    //

    var len = participantList.length;

    bDuplicate = false;
    for (k = 0; k < len; k++)
    {
        participantList.cursor = k;

        if (participantList.USER_ID == memberId)
        {
            bDuplicate = true;
            break;
        }
    }

    if (bDuplicate) continue;

    projectMgr.addParticipant(serviceName,
                              memberId,
                              projectId,
                              contact,
                              owner,
                              contactInfo);

    if (Error.set)
    {
        if (Error.code == 21503) continue;   // BV_Profile_Dup_MValue
        epErrorSystem;
    }
}
```

BVI_EPProjectManager::updateParticipant()

Updates the status of a project participant.

JavaScript

```
long updateParticipant(
    string serviceId,
    long userId,
    string projectId,
    boolean contact,
    boolean owner,
    string contactInfor);
```

Java

```
public final long updateParticipant(
    String serviceId,
    long userId,
    String projectId,
    boolean contact,
    boolean owner,
    String contactInfor);
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	String: ID or name of the current service.
`userId`	Integer; user account ID.
`projectId`	String; ID of the project.
`contact`	Boolean; true means the participant being added is a contact person for the project.
`owner`	Boolean; true means the participant being added is an owner of the project.
`contactInfor`	String; includes information about the participant, such as the participant's title.

Return value

A status code indicating whether the participant has been updated. A value of "0" means success; a value of "1" means failure.

Example

```
projectMgr.updateParticipant(Session.serviceName, memberId, projectId,
contact_r, owner_r, contactInfo);
```

BVI_EPProjectManager::deleteParticipant()

Removes a participant from a project.

JavaScript

```
long deleteParticipant(
    string serviceId,
    long userId,
    string projectId);
```

Java
```
public final long deleteParticipant(
    String serviceId,
    long userId,
    String projectId);
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	String: ID or name of the current service.
`userId`	Integer; user account ID.
`projectId`	String; ID of the project.

Return value A status code indicating whether the participant has been removed from the project. A value of “0” means success; a value of “1” means failure.

Example
```
if (action == "remove")
{
    //
    // Get the ID of the participant to remove.
    //

    var memberId = Request.value("userId");

    if (memberId == null) epErrorMissingBrowserData( 'userId' );

    //
    // Remove the participant from the project.
    //

    projectMgr.deleteParticipant(serviceName, memberId, projectId);

    if (Error.set) epErrorSystem;
}
```

BVI_EPProjectManager::isParticipant()

Verifies a user ID is a participant in a project.

JavaScript
```
boolean isParticipant(
    string serviceId,
    long userId,
    string projectId );
```

Java
```
public final boolean isParticipant(
    String serviceId,
    long userId,
    String projectId );
```

Java exception Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
userId	Integer; user account ID.
projectId	String; ID of the project.

Return value — True if the user ID is a participant of the project, otherwise false.

Example

```
var bParticipant = projectMgr.isParticipant(serviceName, userId,
projectId);

if (Error.set) epErrorSystem;

if (!bParticipant) epErrorMessageId(6808);
```

BVI_EPProjectManager::getParticipantInProject()

Gets all the participants for a project.

JavaScript

```
BVI_ContentList getParticipantInProject(
    string serviceId,
    string projectId);
```

Java

```
public final BVI_ContentList getParticipantInProject(
    String serviceId,
    String projectId);
```

Java exception — Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
projectId	String; ID of the project.

Example

```
var participantList = projectMgr.getParticipantInProject(serviceName,
                                                          projectId);
```

Remarks — Returns a list of participants of the project. The BVI_Content in the list has attributes for USER_ID, EP_PROJECT_CONT, EP_PROJECT_OWNER and EP_CONTACT_INFO.

BVI_EPProjectManager::getContactInProject()

Gets the contact persons related to a project collaboration page.

JavaScript

```
BVI_ContentList getContactInProject(
    string serviceId,
    string projectId);
```

Java
```
public final BVI_ContentList getContactInProject(
    String serviceId,
    String projectId);
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	String: ID or name of the current service.
`projectId`	String; ID of the project.

Return value A BVI_ContentList containing a list of all the contact people for the project.

Example
```
var userList = projectMgr.getContactInProject(serviceName, projectId);
```

Using group methods

The BVI_EPProjectManager group methods enable you to view project groups related to a project collaboration page, view project groups to which a specific visitor belongs, and delete a project group.

Group methods	Functionality
getGroupInProject()	Gets the groups for a specific project.
getGroupForUser()	Gets the groups to which the visitor belongs.
deleteGroup()	Deletes the specified group.

BVI_EPProjectManager::getGroupInProject()

Gets the groups for a specific project.

JavaScript
```
BVI_ContentList getGroupInProject(
    string serviceId,
    string projectId);
```

Java
```
public final BVI_ContentList getGroupInProject(
    String serviceId,
    String projectId);
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	String: ID or name of the current service.
`projectId`	String; ID of the project.

Return value A BVI_ContentList containing a list of all the groups related to the project.

Example
```
var groups = projectMgr.getGroupInProject(serviceId, projectId);
```

BVI_EPProjectManager::getGroupForUser()

Gets the groups to which the visitor belongs.

JavaScript
```
BVI_ContentList getGroupForUser(
    string serviceId,
    long userId,
    string projectId);
```

Java
```
public final BVI_ContentList getGroupForUser(
    String serviceId,
    long userId,
    String projectId);
```

Java exception Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
userId	Integer; user account ID.
projectId	String; ID of the project

Return value A list of groups to which the visitor belongs.

Example
```
var groups = projectMgr.getGroupForUser(serviceName, userId, projectId);
```

BVI_EPProjectManager::deleteGroup()

Deletes the specified group.

JavaScript
```
void deleteGroup(
    string serviceId,
    long userId,
    string groupId);
```

Java
```
public final void deleteGroup(
    String serviceId,
    long userId,
    String groupId);
```

Java exception Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
userId	Integer; user account ID.
groupId	String; ID of the group.

Return value None.

Remarks This method also deletes the group from the tables BV_EP_TASK_GROUP, BV_EP_ANN_GROUP, BV_EP_PROJ_DGROUP, and BV_EP_DFGRP_GROUP.

Example
```
projectMgr.deleteGroup(serviceName, userId, groupId);
```

Using task methods

The BVI_EPProjectManager task methods let you find all tasks related to a project or a phase of a project.

Task methods	Functionality
getTaskInPhase()	Returns the tasks related to a phase in a project.
getTaskInProject()	Returns a list of all tasks related to a project.
getAssignedTaskInPhase()	Returns the assigned tasks related to a phase in the project.
getAssignedTaskInProject()	Returns a list of all the assigned tasks related to the project.

BVI_EPProjectManager::getTaskInPhase()

Returns the tasks related to a phase in a project.

JavaScript
```
BVI_ContentList getTaskInPhase(
    string serviceId,
    long userId,
    string phaseId);
```

Java
```
public final BVI_ContentList getTaskInPhase(
    String serviceId,
    long userId,
    String phaseId);
```

Java exception Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
userId	Integer; user account ID.
phaseId	String; ID of the project phase.

Return value A list of tasks for the specified user ID and phase ID. A visitor can only see tasks if the project group is assigned to that visitor.

Example

```
if (selectedPhaseId == 'all')
{
    cntList = projectMgr.getTaskInProject(serviceName, userId, projectId);
}
else
{
    cntList=projectMgr.getTaskInPhase(serviceName,userId,selectedPhaseId);
}
```

BVI_EPProjectManager::getTaskInProject()

Returns a list of all tasks related to a project.

JavaScript

```
BVI_ContentList getTaskInProject(
    string serviceId,
    long userId,
    string projectId);
```

Java

```
public final BVI_ContentList getTaskInProject(
    String serviceId,
    long userId,
    String projectId);
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
userId	Integer; user account ID.
projectId	String; ID of the project.

Return value

A list of tasks for the specified user ID and project ID. A visitor can only see tasks if the project group is assigned to that visitor.

Example

```
if (selectedPhaseId == 'all')
{
    cntList = projectMgr.getTaskInProject(serviceName, userId,
projectId);
}
else
{
    cntList = projectMgr.getTaskInPhase(serviceName, userId,
selectedPhaseId);
}
```

BVI_EPProjectManager::getAssignedTaskInPhase()

Returns the assigned tasks related to the phase in a project.

JavaScript

```
BVI_ContentList getAssignedTaskInPhase(
    string serviceId,
    long userId,
    string phaseId);
```

Java

```
public final BVI_ContentList getAssignedTaskInPhase(
    String serviceId,
    long userId,
    String phaseId);
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	String: ID or name of the current service.
`userId`	Integer; user account ID.
`phaseId`	String; ID of the project phase.

Return value

A list of assigned tasks for the specified user ID and phase ID. A visitor can only see tasks if the project group is assigned to that visitor.

Example

```
if (selectedPhaseId == 'all')
{
    cntList = projectMgr.getAssignedTaskInProject(serviceName, userId,
projectId);
}
else
{
    cntList = projectMgr.getAssignedTaskInPhase(serviceName, userId,
selectedPhaseId);
}
```

BVI_EPProjectManager::getAssignedTaskInProject()

Returns a list of all assigned tasks related to the project.

JavaScript

```
BVI_ContentList getAssignedTaskInProject(
    string serviceId,
    long userId,
    string projectId);
```

Java

```
public final BVI_ContentList getAssignedTaskInProject(
    String serviceId,
    long userId,
    String projectId);
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	String: ID or name of the current service.
`userId`	Integer; user account ID.
`projectId`	String; ID of the project.

Return value

A list of assigned tasks for the specified user ID and project ID. A visitor can only see tasks if the project group is assigned to that visitor.

Example

```
if (selectedPhaseId == 'all')
{
    cntList = projectMgr.getAssignedTaskInProject(serviceName, userId,
projectId);
}
else
{
    cntList = projectMgr.getAssignedTaskInPhase(serviceName, userId,
selectedPhaseId);
}
```

Using meeting methods

The BVI_EPProjectManager meeting methods let you find all meetings related to a project or a phase of a project.

Meeting methods	Functionality
getMeetingInPhase()	Finds all the meetings related to a phase.
getMeetingInProject()	Finds all the meetings related to a project.

BVI_EPProjectManager::getMeetingInPhase()

Finds all the meetings related to a phase.

JavaScript

```
BVI_ContentList getMeetingInPhase(
    string serviceId,
    long userId,
    string phaseId);
```

Java

```
public final BVI_ContentList getMeetingInPhase(
    String serviceId,
    long userId,
    String phaseId);
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
userId	Integer; user account ID.
phaseId	String; ID of the project phase.

Return value A list of meetings for the specified user ID and phase ID. A visitor can only see tasks if the project group is assigned to that visitor.

Example

```
if (selectedPhaseId == 'all')
{
  cntList = projectMgr.getMeetingInProject(serviceName, userId,
projectId);
}
else
{
  cntList = projectMgr.getMeetingInPhase(serviceName, userId,
selectedPhaseId);
```

BVI_EPProjectManager::getMeetingInProject()

Finds all the meetings related to a project.

JavaScript

```
BVI_ContentList getMeetingInProject(
    string serviceId,
    long userId,
    string projectId);
```

Java

```
public final BVI_ContentList getMeetingInProject(
    String serviceId,
    long userId,
    String projectId);
```

Java exception Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
userId	Integer; user account ID.
projectId	String; ID of the project.

Return value A list of meetings for the specified user ID and project ID. A visitor can only see tasks if the project group is assigned to that visitor.

Example

```
if (selectedPhaseId == 'all')
{
  cntList = projectMgr.getMeetingInProject(serviceName, userId, projectId);
}
else
{
  cntList=projectMgr.getMeetingInPhase(serviceName,userId,selectedPhaseId);
```

Using announcement methods

The BVI_EPProjectManager announcement methods let you find all announcements related to a project or a phase of a project.

Announcement methods	Functionality
getAnnouncementInPhase()	Finds all the announcements related to a phase.
getAnnouncementInProject()	Finds all the announcements related to a project.

BVI_EPProjectManager::getAnnouncementInPhase()

Finds all the announcements related to a phase.

JavaScript

```
BVI_ContentList getAnnouncementInPhase(
    string serviceId,
    long userId,
    string phaseId);
```

Java

```
public final BVI_ContentList getAnnouncementInPhase(
    String serviceId,
    long userId,
    String phaseId);
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
userId	Integer; user account ID.
phaseId	String; ID of the project phase.

Return value

A list of announcements for the specified user ID and phase ID. A visitor can only see tasks if the project group is assigned to that visitor.

Example

```
if (selectedPhaseId == 'all')
{
  cntList = projectMgr.getAnnouncementInProject(serviceName, userId,
        projectId);
}
else
{
  cntList = projectMgr.getAnnouncementInPhase(serviceName, userId,
    selectedPhaseId);
}
```

BVI_EPProjectManager::getAnnouncementInProject()

Finds all the announcements related to a project.

JavaScript

```
BVI_ContentList getAnnouncementInProject(
    string serviceId,
    long userId,
    string projectId);
```

Java

```
public final BVI_ContentList getAnnouncementInProject(
    String serviceId,
    long userId,
    String projectId);
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	String: ID or name of the current service.
`userId`	Integer; user account ID.
`projectId`	String; ID of the project.

Return value

A list of announcements for the specified user ID and project ID. A visitor can only see tasks if the project group is assigned to that visitor.

Example

```
if (selectedPhaseId == 'all')
{
  cntList = projectMgr.getAnnouncementInProject(serviceName, userId,
    projectId);
}
else
{
  cntList = projectMgr.getAnnouncementInPhase(serviceName, userId,
    selectedPhaseId);
}
```

Miscellaneous project methods

The following BVI_EPProjectManager methods don't fit into any other categories yet are important for managing a project collaboration page.

Miscellaneous methods	Functionality
getDgroupInProject()	Finds all the discussion groups the visitor may see in the project.
isOwnerofProject()	Verifies whether the user ID is an owner of the project.
flushContent()	Flush the caches of all running Interaction Managers for a content item.

BVI_EPProjectManager::getDgroupInProject()

Finds all the discussion groups the visitor may see in the project.

JavaScript

```
BVI_ContentList getDgroupInProject(
    string serviceId,
    long userId,
    string projectId);
```

Java

```
public final BVI_ContentList getDgroupInProject(
    String serviceId,
    long userId,
    String projectId);
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
userId	Integer; user account ID.
projectId	String; ID of the project.

Return value

A list of discussion for the specified user ID and project ID.

Example

```
var catList = projectMgr.getDgroupInProject(
    serviceName,
    userId,
    projectId)
```

BVI_EPProjectManager::isOwnerofProject()

Verifies whether the user ID is an owner of the project.

JavaScript

```
boolean isOwnerofProject(
    string serviceId,
    long userId,
    string projectId);
```

Java

```
public final boolean isOwnerofProject(
    String serviceId,
    long userId,
    String projectId);
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
userId	Integer; user account ID.
projectId	String; ID of the project.

Return value

"True" if the user ID is the project owner.

Example

```
var isOwner = projectMgr.isOwnerofProject(
    serviceName,
    userId,
    projectId);

if (!isOwner) epErrorMessageId(6812);
```

BVI_EPProjectManager::flushContent()

Flush the caches of all running Interaction Managers for a content item.

JavaScript

```
void flushContent(
    string serviceId,
    long contentOID,
    long contentType);
```

Java

```
public final void flushContent(
    String serviceId,
    long contentOID,
    long contentType);
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	String: ID or name of the current service.
contentOID	The Object ID of the content item to be flushed.
contentType	The name or ID of the content type, such as EP_Task and EP_Announcement.

Return value None.

Remarks After editing or deleting a task, meeting, or announcement, the script must call flushContent.

Example

```
var projectMgr = Session.epProjectManager;
projectMgr.flushContent(Session.serviceName, contentType,
    contentOid);

if (Error.set) epErrorSystem;
```

BVI_EPProject

The BVI_EPProject object enables you to define a project and its attributes.

```
implementation
{
    include <jsrt/datetime_i.hh>
    include <jsrt/properties_i.hh>
    include <jsrt/valuelist_i.hh>
    include "epproject_i.hh"
}
```

This component does not allow dynamic properties.

The JavaScript component interface files are located in $BV1TO1/include/jsi/bv/webapps; Java classes reside in the com.broadvision.ieportal.components Java packages.

The following table lists the attributes defined by BVI_EPProject.

Attribute	Functionality
projectId	The project ID.
projectName	The project name.
projectDescription	The project description.
projectGoal	The goal of this project.
projectPagepath	The path from the script-root directory to the project page.
projectIconpath	The path from the script-root directory to the icon representing the project on the project collaboration page.
projectCurrentphase	Current phase of the project.
status	The status of the project; 0=off-line, 1=on-line.
strcol1	First extra text data field.
strcol2	Second extra text data field.
strcol3	Third extra text data field.

BVI_EPProject::projectId

The project ID.

JavaScript
```
readonly attribute long projectId;
```

Java
```
public final long getProjectId()
```

Java exception
Throws BVWComponentException.

Example
```
for (i = 0; i < projectListLen; i++)
{
 var item = projectList.get(i);
 var project = item.objectValue;
 var thisProjectId = project.projectId;
 var isOwner = projectMgr.isOwnerofProject(serviceName, userId,
thisProjectId);
}
```

BVI_EPProject::projectName

The project name.

JavaScript
```
attribute string projectName;
```

Java
```
public final String getProjectGoal()
public final void setProjectGoal(String projectGoal)
```

Java exception
Throws BVWComponentException.

Example
```
project.projectName = stringTrim(projectName);
```

BVI_EPProject::projectDescription

The project description.

JavaScript
```
attribute string projectDescription;
```

Java
```
public final String getProjectDescription()
public final void setProjectDescription(String projectDescription)
```

Java exception
Throws BVWComponentException.

Example
```
project.projectDescription = projectDesc;
```

BVI_EPProject::projectGoal

The goal of this project.

JavaScript
```
attribute string projectGoal;
```

Java
```
public final String getProjectGoal()
public final void setProjectGoal(String projectGoal)
```

Java exception Throws BVWComponentException.

Example
```
project.projectGoal = projectGoal;
```

BVI_EPProject::projectPagepath

The path from the script-root directory to the project page. This field is currently unused.

JavaScript
```
attribute string projectPagepath;
```

Java
```
public final String getProjectPagepath()
public final void setProjectPagepath(String projectPagepath)
```

Java exception Throws BVWComponentException.

Example
```
project.projectPagePath = projectPagepath;
```

BVI_EPProject::projectIconpath

The path from the *doc_root* directory to the icon representing the project on the project collaboration page.

JavaScript
```
attribute string projectIconpath;
```

Java
```
public final String getProjectIconpath()
public final void setProjectIconpath(String projectIconpath)
```

Java exception Throws BVWComponentException.

Example
```
project.projectIconpath = isEmptyString(attachIcon) ? projectIcon :
attachIcon;
```

BVI_EPProject::projectCurrentphase

Current phase of the project.

JavaScript
```
attribute long projectCurrentphase;
```

Java
```
public final long getProjectCurrentphase()
public final void setProjectCurrentphase(long projectCurrentphase)
```

Java exception Throws BVWComponentException.

Example
```
if (currentPhaseId != null)
{
    project = projectMgr.getProject(serviceName, projectId);

    if (Error.set) epErrorSystem;

    project.projectCurrentphase = currentPhaseId;
    projectMgr.updateProject(serviceName, userId, project);

    if (Error.set) epErrorSystem;

}
```

BVI_EPProject::status

The status of the content; 0=off-line, 1=on-line.

JavaScript
```
attribute long status;
```

Java
```
public final long getStatus()
public final void setStatus(long status)
```

Java exception Throws BVWComponentException.

BVI_EPProject::strcol1

First extra text data field.

JavaScript
```
attribute string strcol1;
```

Java
```
public final String getStrcol1()
public final void setStrcol1(String strcol1)
```

Java exception Throws BVWComponentException.

Example
```
Project.strcol1 = “ABC”;
```

BVI_EPProject::strcol2

Second extra text data field.

JavaScript
```
attribute string strcol2;
```

Java
```
public final String getStrcol2()
public final void setStrcol2(String strcol2)
```

Java exception Throws BVWComponentException.

Example
```
Project.strcol2 = “ABC”;
```

BVI_EPProject::strcol3

Third extra text data field.

JavaScript
```
attribute string strcol3;
```

Java
```
public final String getStrcol3()
public final void setStrcol3(String strcol3)
```

Java exception Throws BVWComponentException.

Example
```
Project.strcol3 = “XYZ”;
```

BVI_EPProject methods

The following methods are defined by BVI_EPProject:

creator()	Creates a BVI_EPProject object.
creator()	Returns a copy of an existing BVI_EPProject object.
clone()	Returns a copy of this project, or null on error (JavaScript).
clone_J()	Returns a copy of this project, or null on error (Java).

BVI_EPProject::creator()

Creates a BVI_EPProject object.

JavaScript
```
BVI_EPProject creator;
```

Java
```
public BVI_EPProject creator()
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError.

Return value An instance of BVI_EPProject; otherwise an error flag is raised.

BVI_EPProject::creator()

Returns a copy of an existing BVI_EPProject object.

JavaScript
```
BVI_EPProject creator(BVI_EPProject ref);
```

Java
```
public BVI_EPProject creator(ref)
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Return value A copy of a BVI_EPProject object; otherwise an error flag is raised.

BVI_EPProject::clone()

Returns a copy of this object, or null on error.

JavaScript
```
BVI_EPProject clone;
```

Return value A copy of this object, or null on error.

Example
```
var ep_proj = new BVI_EPProject;
var clone1 = ep_proj.clone;
```

BVI_EPProject::clone_J()

Returns a copy of this object, or null on error.

Java
```
public final BVI_EPProject clone_J()
```

Java exception Throws BVWComponentException.

Return value A copy of this object, or null on error.

BVI_EPPhase

The BVI_EPPhase object enables you to define a project phase and its attributes.

```
implementation
{
    include <jsrt/datetime_i.hh>
    include <jsrt/properties_i.hh>
    include <jsrt/valuelist_i.hh>
    include "epphase_i.hh"
}
```

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; Java classes reside in the `com.broadvision.ieportal.components` Java packages.

This component does not allow dynamic properties.

BVI_EPPhase attributes

The following attributes are set by BVI_EPPhase.

Attributes	Functionality
phaseId	The phase ID of this phase
phaseName	The phase name of this phase
phaseDescription	The phase description of this phase
phaseGoal	The phase goal of this phase
projectId	The project ID of the project containing this phase
status	The status of the phase; 0=Off-line, 1=On-Line
strcol1	First extra text data field
strcol2	Second extra text data field
strcol3	Third extra text data field

BVI_EPPhase::phaseId

The phase ID of this phase.

JavaScript

```
readonly attribute long phaseId;
```

Java

```
public final long getPhaseId()
```

Java exception

Throws BVWComponentException.

Example

```
for (i = 0; i < len; i++)
{
    phaseList.cursor = i;

    var phase = phaseList.objectValue;
    var name = phase.phaseName;
    var desc = phase.phaseDescription;
    var phaseId = phase.phaseId;
}
```

BVI_EPPhase::phaseName

The phase name of this phase.

JavaScript
```
attribute string phaseName;
```

Java
```
public final String getPhaseName()
public final void setPhaseName(String phaseName)
```

Java exception
Throws BVWComponentException.

Example
```
var phase = new BVI_EPPhase;

phase.phaseName = stringTrim(phaseName);
phase.phaseDescription = phaseDesc;
phase.phaseGoal = phaseGoal;
phase.projectId = projectId;
phase.status = 1;   // 0="Off-line", 1="On-line"
```

BVI_EPPhase::phaseDescription

The phase description of this phase.

JavaScript
```
attribute string phaseDescription;
```

Java
```
public final void setPhaseDescription(String phaseDescription)
public final String getPhaseDescription()
```

Java exception
Throws BVWComponentException.

Example
```
var phase = new BVI_EPPhase;

phase.phaseName = stringTrim(phaseName);
phase.phaseDescription = phaseDesc;
phase.phaseGoal = phaseGoal;
phase.projectId = projectId;
phase.status = 1;   // 0="Off-line", 1="On-line"
```

BVI_EPPhase::phaseGoal

The phase goal of this phase.

JavaScript
```
attribute string phaseGoal;
```

Java
```
public final String getPhaseGoal()
public final void setPhaseGoal(String phaseGoal)
```

Java exception Throws BVWComponentException.

Example
```
var phase = new BVI_EPPhase;

phase.phaseName = stringTrim(phaseName);
phase.phaseDescription = phaseDesc;
phase.phaseGoal = phaseGoal;
phase.projectId = projectId;
phase.status = 1;   // 0="Off-line", 1="On-line"
```

BVI_EPPhase::projectId

The project ID of the project containing this phase.

JavaScript
```
attribute long projectId;
```

Java
```
public final long getProjectId()
public final void setProjectId(long projectId)
```

Java exception Throws BVWComponentException.

Example
```
var phase = new BVI_EPPhase;

phase.phaseName = stringTrim(phaseName);
phase.phaseDescription = phaseDesc;
phase.phaseGoal = phaseGoal;
phase.projectId = projectId;
phase.status = 1;   // 0="Off-line", 1="On-line"
```

BVI_EPPhase::status

The status of the phase; 0=Off-Line, 1=On-Line.

JavaScript
```
attribute long status;
```

Java
```
public final long getStatus()
public final void setStatus(long status)
```

Java exception Throws BVWComponentException.

Example
```
var phase = new BVI_EPPhase;

phase.phaseName = stringTrim(phaseName);
phase.phaseDescription = phaseDesc;
phase.phaseGoal = phaseGoal;
phase.projectId = projectId;
phase.status = 1;   // 0="Off-line", 1="On-line"
```

BVI_EPPhase::strcol1

First extra string data field.

JavaScript
```
attribute string strcol1;
```

Java
```
public final String getStrcol1()
public final void setStrcol1(String strcol1)
```

Java exception Throws BVWComponentException.

Example
```
phase.strcol1 = “ABC”;
```

BVI_EPPhase::strcol2

Second extra string data field.

JavaScript
```
attribute string strcol2;
```

Java
```
public final String getStrcol2()
public final void setStrcol2(String strcol2)
```

Java exception Throws BVWComponentException.

Example
```
phase.strcol2 = “XYZ”;
```

BVI_EPPhase::strcol3

Third extra string data field.

JavaScript
```
attribute string strcol3;
```

Java
```
public final String getStrcol3()
public final void setStrcol3(String strcol3)
```

Java exception Throws BVWComponentException.

Example
```
phase.strcol3 = “XYZ”;
```

BVI_EPPhase methods

The following methods are part of BVI_EPPhase.

creator()	Creates an instance of BVI_EPPhase.
creator()	Returns a copy of an existing BVI_EPPhase object.
clone()	Copies the specified BVI_EPHPhase object (JavaScript).
clone_J()	Copies the specified BVI_EPHPhase object (Java).

BVI_EPPhase::creator()

Creates an instance of a BVI_EPPhase object.

JavaScript
```
BVI_EPPhase creator;
```

Java
```
public BVI_EPPhase creator()
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Return value An instance of BVI_EPPhase; otherwise an error flag is raised.

BVI_EPPhase::creator()

Returns a copy of an existing BVI_EPPhase object.

JavaScript
```
BVI_EPPhase creator(BVI_EPPhase ref);
```

Java
```
public BVI_EPPhase creator(BVI_EPPhase ref)
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Return value A copy of a BVI_EPPhase object; otherwise an error flag is raised.

BVI_EPPhase::clone()

Copies the specified BVI_EPPhase object.

JavaScript
```
BVI_EPPhase clone();
```

Return value A copy of this object, or null on error.

BVI_EPPhase::clone_J()

Copies the specified BVI_EPPhase object.

Java `public final BVI_EPPhase clone_J()`

Java exception Throws BVWComponentException.

Return value A copy of this object, or null on error.

Using the project collaboration page wizard

InfoExchange Portal has a project collaboration wizard you can include in your customized page. Visitors can use this wizard to create a project collaboration page.

To hook the wizard into an application, you need to call `pw_newProject.jsp`, which is the first page of the wizard. To set up the hook, make sure the visitor is a Project Administrator, such as in this example taken from the BVAdventech sample application:

```
<% if (projectAdmin == 1) { %>

<a href="#" onclick=newWindow><img alt="Create New Project" border=0
height=45 hspace=0 src="/adventech/images/pm_newProject.gif"
width=145></a><br>

<% } else { %>

<img src='/adventech/images/pm_myProjects.gif' width='145' height='45'
border='0' alt='My Projects'><br>

<% } %>
```

The client-side JavaScript function `newWindow` opens the project wizard in a pop-up window by calling `pw_newProject.jsp`:

```
function newWindow
{
    window.open( <%= dquote(
      makeScriptURL(
"/adventech/scripts/project/wizard/pw_newProject.jsp" ) )
%>,
      "wizard","resizable=yes,scrollbars=yes,height=550,width=525");
}
```

Project discussion groups

Discussion groups allow the project participants to freely exchange ideas. There are two types of discussion groups:

- private
- public

Files can be attached to posted messages. The messages in the discussion groups are threaded. The BVI_DForumManager component, described in the *Developer's Guide to Components and Scripts*, enables you to manage discussion groups.

Creating and deleting discussion groups

When you create a project, the project collaboration wizard automatically creates a Public discussion group open to all participants in the collaboration. A project owner can add more discussion groups by selecting **Add Discussion**. If you delete a discussion group or message, you need to delete the attached filename in the database as well as the actual attached files.

The `$BV1TO1/erm_app/examples/adventech/scripts/project/discussions.jsp` script contains an example of deleting a discussion group and its attached filenames from the database.

The `$BV1TO1/erm_app/examples/adventech/scripts/project/deleteMessages.jsp` script contains an example of deleting a message and its attached filenames from the database. See the *InfoExchange Portal Administrator's Guide* for information on deleting the actual attached files.

Discussion group attachments

The scripts `viewDiscussionMsg.jsp`, `attach_util.js`, and `postMessage.jsp` are used by InfoExchange Portal to handle discussion groups.

viewDiscussionMsg.jsp

The script `viewDiscussionMsg.jsp` collects the discussion message and filenames. It then uploads the files and submits them to `cp_upload.exe`, which puts the files on the HTTP server, and then redirects back to the script `viewDiscussionMsg.jsp` with the action "`actionPost`", along with the filenames and file sizes of the discussion message and filenames.

This script handles the "`actionPost`" action by storing the filenames and sizes in the `BV_DF_FILES` table.

The script `viewDiscussionMsg.jsp` lets you:

- display a discussion group message for a project
- display the messages for the discussion group
- enable a participant to reply to a discussion group message
- enable a participant to attach additional files to a message

The `viewDiscussionMsg.jsp` script supports a single form attachment using `cp_upload.exe`. All values, including any attachments, are initially submitted to `cp_upload.exe` using a multi-part form post.

After parsing out any attached files, `cp_upload.exe` redirects the remaining attribute/value pairs to "`nextPage`", which is set to this page by the `epAttachHiddenFieldsAsString` method in `attach_util.js`.

The `cp_upload.exe` program returns the path of the upload file relative to the *`doc-root`* as well as the size of the uploaded file in bytes. For example:

```
<input type=filename=attachFileN>
```

As part of the redirect, `cp_upload.exe` returns:

```
attachFileN = "/relativeDir/filename"
attachFileN_size = file_size
```

Attachments in the BVAdvenTech sample application are uploaded to:

`doc-root``/bv_dgattachments/`*`groupId`*

where *`groupId`* is the OID of the discussion group for the message. For example:

`doc-root``/bv_dgattachments/8234`

The discussion group OID is set by the `epAttachHiddenFieldsAsString(nextPage, relativeDir)` method.

The first parameter to `epAttachHiddenFieldsAsString` specifies `nextPage`, the page to which `cp_upload.exe` redirects; it defaults to self.

The second parameter specifies `relativeDir`, the directory in which the uploaded files are placed relative to *`doc-root`*.

```
<%=
epAttachHiddenFieldsAsString(makeScriptURL('/adventech/scripts/project/vi
ewDiscussionMsg.jsp'), "bv_dgattachments/" + groupId) %>
...
<td><input type="submit" name="actionPost" value="Post Message"
onClick="return validateForm()"></td>
```

The following code sets the action to the upload program *`doc-root`*`/cgi-bin/cp_upload.exe` and sets the encoding type to `multipart/form-data`:

```
<form action="/cgi-bin/cp_upload.exe" name="attachForm" method="post"
enctype="multipart/form-data">
```


See the *Instant Publisher* documentation for more information on `cp_upload.exe`.

attach_util.js

The script `attach_util.js` creates the hidden fields required by `cp_upload.exe` utility using the function `epAttachHiddenFieldsAsString(`*`targetURL, relativeDir`*`)` where:

targetURL	is the URL to which `cp_upload.exe` redirects after processing any attachments
relativeDir	is the location into which attachments are placed relative to *`doc-root`*

The `cp_upload.exe` utility accepts the following attributes:

nextPage	the URL to which `cp_upload.exe` redirects after processing any attachments
app_name	the name of the Interaction Manager
relative_dir	the location into which attachments are placed relative to *`doc-root`*
upload_option	`rename_file`: Generate a unique filename if the specified file already exists in *relative_dir.* `overwrite_file`: Overwrite a file if the specified file already exists in *relative_dir*

This function returns a concatenated string consisting of the required attributes for `cp_upload.exe` formatted as HTML hidden input fields.

postMessage.jsp

The script `postMessage.jsp` posts a message to a discussion group; it is run when the action is "`actionPost`".

```
var filename = Request.value("attachFile");
var size = Request.value("attachFile_size");
var cntMgr = new BVI_ContentManager("-1");
cntMgr.represent(currentVisitor());

if ( !isEmptyString(filename) )
    {
    //
    // Create the attachment for the message.
    //

    var row = new BVI_MultiValueRow("BV_DF_FILES", Session.serviceName,
"DF_MESSAGE");
    var value = new BVI_Value;

    value.stringValue = filename;
    row.set("FILE_PATH", value);

    value.longValue = size;
    row.set("FILE_SIZE", value);


    var attrList = new BVI_StringList;

    attrList.append("FILE_PATH");
    attrList.append("FILE_SIZE");


    //
    // The value of messageID is the message you want to attach to.
    //
    cntMgr.insertMultiValueRow(row, messageID, attrList);

    if (Error.set)
    {
        epErrorSystem();
    }

}
```

6 Customizing Closed-loop Process Management

This chapter describes the features and usage of the closed-loop process management feature of InfoExchange Portal.

The following topics are covered in this chapter:

Introduction

Closed-loop process management changes the collaborative nature of enterprise relationship management by automatically tracking specific uses of an enterprise portal. The InfoExchange Portal closed-loop process management system can be used to track and manage action items, ticket items, or leads. In most implementations, tickets and action items are generated through the collaboration page. For example, ticket items can be issues discovered during a meeting or posted on the collaboration page, and action items can be defined steps within a phase of a project. Leads however, are generated from the enterprise web site when a visitor requests more information about a particular product, such as a white paper, product demonstration, product sample, or a test drive.

A new item, such as an action, ticket, or lead, is created using an instant publisher form. Items advancing through the closed-loop process are never lost, as the system tracks the progress and the individuals involved. A manager can quickly see which items are new, resolved, unresolved, or deleted. If an individual to whom an item has been assigned cannot resolve the issue or complete the task, the manager can assign the item to someone else.

Sample closed-loop process implementation

InfoExchange Portal ships with a closed-loop process management example usage of the closed-loop process management. In this example the host corporation may institute a closed-loop process to track and distribute leads gathered from potential customers clicking through the enterprise web site. Leads are then automatically assigned to appropriate channel partners and their status continually tracked until the desired end result is achieved or the lead is manually terminated by the managing individual.

Closed-loop process management enables leads, such as sales leads or bugs being tracked, to be created after a visitor accesses a site. When a visitor requests information from a site, he or she fills in an instant publisher form to provide background information. Submitting the form creates a lead content item which is initially routed to a lead manager within the company running the site, and then to a lead owner.

Unlike the other parts of InfoExchange Portal, there is no specialized API for closed-loop process management. Instead, you modify scripts supplied with InfoExchange Portal to customize closed-loop process management. These scripts use One-To-One Enterprise and Instant Publisher APIs.

For example, you can customize the instant publisher forms that generate leads to prompt the visitor for the custom lead attributes. You can also customize the alert script that runs the rule set that selects the correct reseller or sales person.

The *lead owner organizations* are the organizations that can own leads, such as a reseller. The sales manager is an employee of the host enterprise who manages the lead owner organizations. Each lead owner organization has a contact person. Leads typically are assigned to the contact person of a lead owner organization. The sales manager can reassign leads. Rules and alerts automatically assign new leads to the contact person, but the sales manager can override these rules to assign the lead to a different contact person.

A *sales agent* is a lead owner organization, and leads are assigned to a person (called a *contact*) and not to the owner organization (sales agent).

InfoExchange Portal automatically sends e-mail that:

- notifies the contact person when a lead is assigned to them
- reminds the contact person to accept or decline the lead
- notifies the sales manager when a lead is automatically rerouted to the manager because the contact person did not act on the lead after a specified period of time


In the BVAdvenTech sample application, the lead owner organization is referred to as the sales agent.

The closed-loop process management scripts for the BVAdvenTech sample application are located in *script-root*`/adventech/scripts/leads`. All the alert scripts for closed-loop process management are located in `$BV1TO1_VAR/msg_scripts`.

Creating a lead

A new LEAD content type item being created and added to the BV_EP_LEAD table constitutes a lead. Using the instant publisher APIs an initial "contact me" form makes the entry into the database and starts the flow process. The contact_me.jsp form initializes the lead by setting the following values in the BV_EP_LEAD table:

- ZIP - the matching alert matches against this value to the zip codes, located in BV_EP_ORG_ZIP, covered by a sales agent
- STATUS - the lead's status must be "On-line" (or 1) for the matching alert to act on it.
- HRS_TO_AUTO_ASSIGN - the number of hours to wait after finding a sales agent to handle this lead and the time of actually changing the lead assignment from the owner organization manager to the sales agent. The default is 8 hours.
- HRS_TO_NEXT_REMIND - the number of hours to wait between sending another reminder e-mail to the assigned sales agent's contact verifying the assignment. The default is 24 hours.
- REMINDER_COUNTER - the number of times to remind a sales agent's contact they have been assigned this lead before reassigning it back to the owner organization manager. The default is 3 times.

If you want the value of REMINDER_COUNTER to reset to a value other than 3, you must also change the value in lead_detail.jsp to match.

- PRODUCT - the matching alert matches against this value to the products, located in BV_EP_ORG_PRODUCT, covered by a sales manager
- DEF_OWNER_ID, MANAGER_ID, CP_WORKFLOW_STATE - the matching alert determines and sets these values.

The "contact me" form must initialize these values to "0" for the matching alert to work.

This form also sets all of the custom personal attributes of the lead, such as the name of the prospect (NAME), the quantity desired (QUANTITY), and the comments to the sales agent (COMMENT_FOR_OWNER).

The last thing the contact_me.jsp form does is make an entry in the bv_alert_spec table so the first alert (the matching alert) can start the process.

Alert #1 - $BV1TO1_VAR/msg_scripts/ep_matching_alert.jsp

This alert, like all other One-To-One Enterprise alerts, has been configured to activate at an interval defined by the visitor. When the alert fires, it gets a list of leads to act on from the bv_alert_spec table.

This script by default calls the multi-value attribute matching rule Find Owner Org.

By default the script compares the values of the BV_EP_LEAD.PRODUCT attribute with the BV_EP_ORG_PRODUCT table to get a list of owner organizations that specialize in that product. The script then compares the BV_EP_LEAD.ZIP attribute with the BV_EP_ORG_ZIP table to get a list of owner organizations that claim coverage of that zip code. Next, the script compares the BV_EP_LEAD.COUNTRY attribute with BV_EP_OWNER_ORG.COUNTRY to get a list of owner organizations in that country. This list of owner organizations are then AND matched together to find only owner organizations that match the product, the zip code, and the country in the

BV_EP_LEAD table. If there are multiple matches, the first owner organization is selected; if there are no owner organizations, a random owner organization is selected by the script ep_matching_alert.jsp.

Once the owner organization has been determined, the alert updates the following attributes in the BV_EP_LEAD table for that lead:

- DEF_OWNER_ID - sets this value to the ID of the contact person of the owner organization matched.
- DEF_OWNER_ORG_OID - sets this value to the OID of the matched owner organization.
- MANAGER_ID - sets this value to the manager of the matched owner organization.
- CP_WORKFLOW_STATE - sets the workflow state to "New."
- CP_ASSIGNED_TO - assigns the lead to the manager of the owner organization (MANAGER_ID)
- NEXT_SCHED_TIME - the exact time to auto-assign this lead is calculated and updated into this column for use by the next alert (the assignment alert). The current time is retrieved, and the value in HRS_TO_AUTO_ASSIGN is added. The resulting date and time sequence is then set in this attribute

An entry is then made in the history table.

Note that this alert does not actually assign the lead to the sales agent's contact (the contact person for the owner organization) determined from the match. Instead, it assigns the lead to the manager of the owner organization to give the manager some time to determine whether to assign the lead to someone other than the contact for the sales agent determined by the matching rule.

Finally, this alert removes the entry for the first alert for this lead from the bv_alert_spec table and makes a new entry for this lead for the second alert (the assignment alert) to continue the process of the lead through the workflow.

All the following alerts work the same way; when the alert is called, One-To-One Enterprise executes a complex SQL statement. It is part of the alert and supplied with the alert implementation. To see the specific SQL calls, see the file:

$BV1TO1/common_erm/alerts/leads/BV_ALERT_TYPES.*xxx*.dmp

where *xxx* is ora for Oracle, syb for Sybase, and inf for Informix. This alert finds all content that applies to the alert in the alert_spec table and then runs the SQL code to filter which of them to process this time around.

Alert #2 - $BV1TO1_VAR/msg_scripts/ep_assignment_alert.jsp

This alert, like all other One-To-One Enterprise alerts, has been configured to activate at an interval defined by the visitor. When the alert fires, it gets a list of leads to act on from the bv_alert_spec table.

This lead actually does the assignment to the matched sales agent's contact. If the current time is after the time in NEXT_SCHED_TIME, it is past the time to wait set in HRS_TO_AUTO_ASSIGN, and the following attributes for this lead in the BV_EP_LEAD table are updated:

- CP_WORKFLOW_STATE - updates to "Assigned"
- CP_ASSIGN_TO - sets to the contact person of the sales agent determined by the matching alert, which was previously stored in DEF_OWNER_ID by the script ep_matching_alert.jsp.

- OWNER_ORG_OID - sets to the OID of the owner organization (the sales agent). This is the same as the DEF_OWNER_ORG_OID determined by the matching rules in the script ep_matching_alert.jsp
- NEXT_SCHED_TIME - the exact time to next remind the assigned sales agent's contact is calculated and updated into this column for use by the next alert (the reminder alert). The current time is retrieved, and the value in HRS_TO_NEXT_REMIND is added. The resulting date and time sequence is then set in this attribute.

An entry is made in the history table, and an e-mail message is then sent to the e-mail address of the newly assigned contact person for the sales agent recorded in the profile as the newly assigned lead. The subject and body of the message are defined in the script ep_assignment_alert.jsp.

Finally, this alert removes the alert entry for this lead in the bv_alert_spec table and makes a new entry for this lead for the third and final alert (the reminder alert) to continue the workflow process of the lead.

The SQL filter for this alert looks for all leads with CP_WORKFLOW_STATE set to "New" where the current time is newer than NEXT_SCHED_TIME and passes on others that do not match. However, you must manually remove an "assigned" lead from the alert_spec table to keep it from being considered each time the alert fires in the future. For example, if you create a script that moves a lead out of the "New" state and does not need this assignment alert to act on it any more, the script must manually remove this lead from the bv_alert_spec table, such as when a manager manually assigns a new lead before the auto-assignment alert for the lead has run. The manager uses the lead_details.jsp script to do the assignment, and logic exists in that script to remove the lead from the bv_alert_spec table.

Alert #3 - $BV1TO1_VAR/msg_scripts/ep_reminding_alert.jsp

This alert, like all other One-To-One Enterprise alerts, has been configured to activate at an interval defined by the visitor. When the alert fires, it gets a list of leads to act on from the bv_alert_spec table. The goal of this alert is to remind the assigned sales agent's contact multiple times via e-mail they have a new lead they have not yet acted upon. If the sales agent's contact has been reminded the maximum number of times and has not accepted the lead, the lead is moved back to the manager with the state set to AutoDeclined.

If the following are all true:

- the REMINDER_COUNTER value (originally set in the contact_me.jsp script) is greater than zero
- the current time is after NEXT_SCHED_TIME
- the e-mail attribute is set in the profile of the sales agent's contact

An e-mail message is sent to the contact person (sales agent) defined in CP_ASSIGNED_TO to remind them they have a new lead. The subject and body of the e-mail can be set in the script for this alert, ep_reminding_alert.jsp. If the alert does send an e-mail message, the following attributes are updated in the BV_EP_LEAD table for this lead:

- REMINDER_COUNTER - this value is decremented by 1.
- NEXT_SCHED_TIME - this value is updated to the next time a reminder is to go out. This value is determined by taking the current time and adding the value of HRS_TO_NEXT_REMIND.

The following are exceptions to this:

- If the e-mail attribute is null, no e-mail is sent, but the alert acts as if it has fired and continues the previously described logic.

- If the REMINDER_COUNTER hits zero, this means the agent has been reminded the maximum times and has not acted on the lead. The lead is reassigned back to the manager of the owner organization and the state is moved to "AutoDeclined."

The following attributes are modified in the BV_EP_LEAD table for this lead:

- CP_ASSIGNED_TO - assigned to the value of MANAGER_ID
- CP_WORKFLOW_STATE - set to "AutoDeclined"

When a lead is AutoDeclined, an entry is made to the history table to reflect the reassignment back to the manager and the AutoDeclined state. An e-mail message about this fact is also sent to the manager.

Finally, this alert removes the entry for this lead in the bv_alert_spec table to complete the workflow of the lead.

The SQL filter for this alert looks for leads in the "Assigned" state. If a custom script is written to accept a lead, it must manually take the lead out of the bv_alert_spec table whether or not it changes the CP_WORKFLOW_STATE from "Assigned" so the alert does not consider the lead in the future. For example, the BVAdventech reference application defines "accepted" as a sales manager editing the lead and moving the state to "Accepted." The lead_details.jsp script is where this happens; logic exists in that script to remove the lead from the bv_alert_spec table.

Database tables

The closed-loop process management component of InfoExchange Portal uses the following database tables.

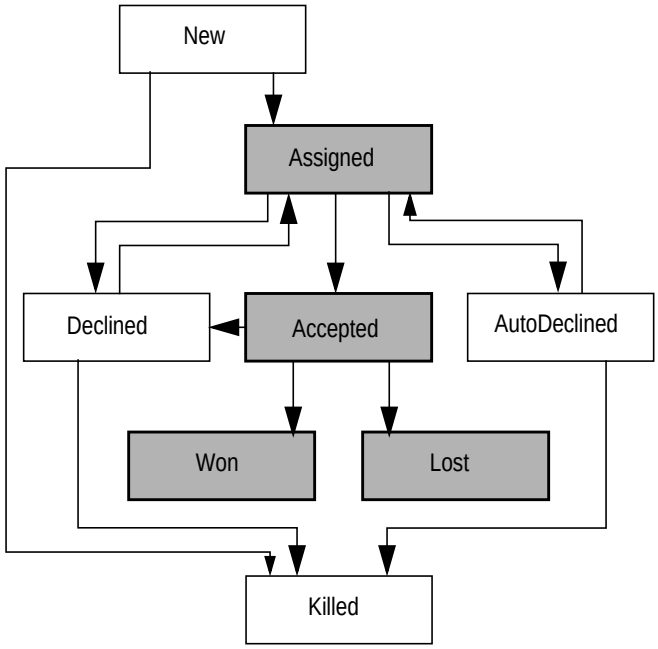
Database table name	Description
BV_EP_LEAD	Leads—stores all the leads in the system
BV_EP_OWNER_ORG	Lead Owner Organizations—stores all the information about the owner organizations in the system, including the identity of the contact person
BV_EP_ORG_PRODUCT	Products Covered by Lead Owner Organization—list table of BV_EP_OWNER_ORG. It stores all the products that are covered by the owner organizations. Each owner organization can cover one or more products. There do not need to be corresponding entries in the table. This is a table in the BVAdvenTech sample application and is not created if you do not load the BVAdvenTech data

Database table name	Description
BV_EP_ORG_ZIP	Zip Codes Covered by Lead Owner Organizations—list table of BV_EP_OWNER_ORG. It stores all the zip codes that are covered by the owner organizations. Each owner organization can cover one or more zip codes in the BV_EP_OWNER_ORG table. This is a table in the BVAdvenTech sample application and is not created if you do not load the BVAdvenTech data
BV_EP_LEAD_HIS	Lead History—stores the history of each lead. A history for a lead includes the previous and current owner of the lead as well as the current workflow state. An entry is entered into this table whenever a script changes the owner or the workflow state of a lead. It is the responsibility of the script to record events in the lead history table
BV_EP_UPROF	Attributes—not exclusively used by closed-loop process management but belongs to the entire InfoExchange Portal site. Closed-loop process management is only concerned with the EP_OWNER_ADMIN attribute. The attribute specifies whether a user is a lead owner administrator

The tables BV_EP_LEAD, BV_EP_OWNER_ORG, BV_EP_ORG_PRODUCT, and BV_EP_ORG_ZIP are BroadVision content types and list tables. Both BV_EP_LEAD and BV_EP_OWNER_ORG are extended with Publishing Center attributes.

Using workflow to manage leads

The following diagram shows the workflow state diagram for closed-loop process management in the BVAdvenTech sample application.



The workflow states in the black boxes are owned by the contact person for the sales agent, while the workflow states in the gray boxes are owned by sales manager.

Changing workflow requires updating the files `lead_detail.jsp` and `lead_inbox.jsp`. You can change the names of the labels of any of the workflow states and transitions "New," "Assigned," and "AutoDeclined." To change the names of these states, copy the following files from the `$BV1TO1/common_erm/alerts/leads` directory to a temporary directory, such as `/tmp`, and then edit them:

- `BV_ALERT_TYPES.xxx.dmp`, where *xxx* is either `ora`, `inf`, or `syb`, depending on which database you are using
  a. Change all occurrences of `'New'` to `'NewNew'` where "*NewNew*" is your replacement for the "New" state text.
  b. Change all occurrences of `'Assigned'` to `'NewAssigned'` where "*NewAssigned*" is your replacement for the "Assigned" state text.
- `ep_matching_alert.jsp`

  Change all occurrences of `"New"` to `"NewNew"` where "*NewNew*" is your replacement for the "New" state text.
- `ep_assignment_alert.jsp`

  Change all occurrences of `"Assigned"` to `"NewAssigned"` where "*NewAssigned*" is your replacement for the "Assigned" state text.
- `ep_reminding_alert.jsp`

  Change all occurrences of `"AutoDeclined"` to `"NewAutoDeclined"` where "*NewAutoDeclined*" is your replacement for the "AutoDeclined" state text.

After you have made the changes to these files, execute the following commands to remove the old alert types and to create the new alert types:

```
cd $BV1TO1/common_erm/alerts/leads
$BV1TO1/bin/scripts/apply_sql del_atypes.xxx.sql
cd /tmp
mv BV_APERT_TYPES.xxx.dmp BV_ALERT_TYPES.dmp
$BV1TO1/bin/bv_load_tbl -T BV_ALERT_TYPES
rm -f /tmp/BV_ALERT_TYPES.dmp
```

Then copy the three script files (`ep*alert.jsp`) to the `$BV1TO1_VAR/msg_scripts` directory.

Using matching rule sets

The goal of closed-loop process management is to match an incoming lead with the appropriate reseller (now called a "sales agent" in the BVAdvenTech user interface, and called "lead owner organization" in the schema).

However, a sales agent is a content item in the `BV_EP_OWNER_ORG` content type, and One-To-One Enterprise doesn't support matching content (`BV_EP_LEAD`) to content (`BV_EP_OWNER_ORG`). Instead, One-To-One Enterprise supports matching content to visitors.

To work around this limitation, the InfoExchange Portal file `ep_matching_alert.jsp` does the following:

1. Creates a temporary guest user for each lead
2. Automatically copies the attributes of the lead ZIP, PRODUCT, and COUNTRY into the profile of the transient guest.

3. Runs a matching rule to match the temporary visitor to a sales agent. The rule must only refer to profile attribute values copied from the lead and not to the other profile attributes.

The script `ep_assignment.jsp` then assigns the lead to the contact person for the matching sales agent. Note that the matching is done by one alert; assignment is done later by another alert.

Understanding the lead history table (BV_EP_LEAD_HIS)

The following example explains the entries made into the lead history table for different events.

LEAD_OID	PREV_OWNER_ID	CURR_OWNER_ID	PREV_OWNER_ORG_OID	CURR_OWNER_ORG_OID	LEAD_STATE	LOG_TIME
101	0	0	0	0	0	t1
101	0	salesmgr	0	0	new	t2
101	salesmgr	agent1's contact	0	agent1	assigned	t3
101	agent1's contact	salesmgr	agent1	0	auto-declined	t4
101	salesmgr	agent2's contact	0	agent2	assigned	t5
101	agent2's contact	agent2's contact	agent2	agent2	accepted	t6
101	agent2's contact	agent2's contact	agent2	agent2	won	t7

Although the actual values in the database for PREV_OWNER_ID, CURR_OWNER_ID, PREV_OWNER_ORG_OID, CURR_OWNER_ORG_OID and LEAD_STATE are numbers instead of text, they are shown as text here for ease of discussion.

1. At t1, a new lead is created.
2. At t2, the matching alert script `ep_matching_alert.jsp` determines the best agent to work on the lead and assigns the lead to that agent's manager. At this point salesmgr can either wait for the lead to be automatically assigned within a certain time (specified by `hours_to_auto_assign` in the script *`script-root`*`/adventech/scripts/leads/contact_me.jsp`) or manually assign the lead to an agent's contact person.
3. At t3, the lead is assigned to agent1's contact person. This can be in response to the assignment alert or the manual assignment by salesmgr.
4. At t4, the lead is assigned back to the salesmgr because agent1's contact person hasn't accepted the lead after the specified number of reminders. This reassignment is made by the reminder alert script `ep_reminder_alert.jsp`.
5. At t5, the lead is assigned to agent2's contact person by salesmgr.
6. At t6, the lead is accepted by agent2's contact person.
7. At t7, agent2's contact person closes the deal.

The BVAdvenTech closed-loop process management scripts record an entry in the lead history table each time a lead is reassigned or the workflow state of a lead changes. If you add or modify the lead management scripts, you need to insert records into the BV_EP_LEAD_HIS lead history table using the One-To-One Enterprise GenericDBManager.

Types of inboxes

By default, sales managers see the following types of inboxes:

- **New** - shows all leads where the `MANAGER_ID` attribute of the lead is the sales manager's user account and `CP_WORKFLOW_STATE` of the lead is `New`.
- **Unassigned** - shows all leads that have not been assigned to anyone. These are newly entered leads that have not yet been assigned by the EP Matching Alert.
- **Declined** - shows all leads where `MANAGER_ID` is the sales manager's user account and `CP_WORKFLOW_STATE` is either `Declined` or `Auto-Declined`
- **All** - shows all leads where `MANAGER_ID` is the sales manager's user account regardless of the value of `CP_WORKFLOW_STATE`

The contact people of sales agents see two types of inboxes:

- **Assigned** - shows all the leads where `CP_ASSIGNED_TO` is the user account of the contact person of the sales agent and `CP_WORKFLOW_STATE` is `Assigned`
- **All** - shows all the leads where `CP_ASSIGNED_TO` is the user account of the contact person of the sales agent regardless of the value of `CP_WORKFLOW_STATE`

The script *`script-root`*`/adventech/scripts/leads/lead_inbox.jsp` runs different queries to create these different kinds of inboxes. You can customize these queries to create other kinds of inboxes.

Using closed-loop process management without the sample application

InfoExchange Portal is shipped with scripts you can modify to customize closed-loop process management for your own application. These scripts are hard-coded for the BVAdvenTech sample application. To make these scripts work with your own application:

- Change the files *`script-root`*`/adventech/scripts/leads/contact_me.jsp` and *`script-root`*`/adventech/scripts/leads/login_adventech.jsp` to use the name of your service instead of "Adventech".
- Add your own contact users and sales agents.
- Modify workflow for your own application.

Modifying the scripts to work without the sample application

To edit or add agents without installing sample data, you need to modify the script `agent.jsp` as follows:

1. Remove the lines that refer to the sample category name `/All` and replace it with your own categories.

   Category `/All` is not created unless the sample data is loaded. Two lines in `agent.jsp` refer to category `/All`.

2. Remove references to the sample schema database tables BV_EP_ORG_PRODUCT and BV_EP_ORG_ZIP and replace them with your own database tables.

   The database tables BV_EP_ORG_PRODUCT and BV_EP_ORG_ZIP are not created unless the sample data is loaded. You therefore need to remove the script code tables that access those two database tables from the agent.jsp form.

3. Add your custom attributes from your closed-loop process management schema.

To make the file contact_me.jsp work without the BVAdvenTech sample data:

1. Replace the service name “Adventech” with the name of your service.

2. Modify the following sample attributes specific to the BVAdvenTech sample application:

- FAX_NUMBER
- PHONE_NUMBER
- BUSINESS_NAME
- COUNTRY
- ZIP
- PRODUCT
- QUANTITY

Either replace these attributes with your custom attributes in BV_EP_LEAD or any list tables associated with BV_EP_LEAD, or remove the lines that refer to them.

To make the file lead_inbox.jsp work without the BVAdvenTech sample data, modify the PRODUCT attribute, as it is specific to the BVAdvenTech sample application. Either replace PRODUCT with your custom attributes in BV_EP_LEAD or any list tables associated with BV_EP_LEAD, or remove the lines that refer to it.

To make the file lead_detail.jsp work without the BVAdvenTech sample data, modify the following sample attributes specific to the BVAdvenTech sample application:

- FAX_NUMBER
- PHONE_NUMBER
- BUSINESS_NAME
- COUNTRY
- ZIP
- PRODUCT
- QUANTITY

Either replace these attributes with your custom attributes in BV_EP_LEAD or any list tables associated with BV_EP_LEAD, or remove the lines that refer to them.

Modifying workflow to work without the sample application

The sample scripts shipped with InfoExchange Portal can be customized for your application. You can rename any workflow states, and you can add and delete workflow states.

CAUTION You must not delete the New, Assigned, and AutoDeclined states because the alert scripts rely on these states.

You can modify the workflow in the following files in *script-root*/adventech/scripts/leads as shown in these examples:

- Change the names of state "New" to "Just In", and the name of the "New" inbox link to "This Just In":
  - In leadNav.jsp, change the hardcoded URL to 'InboxType="Just In".'
  - In lead_inbox.jsp:
    - Change the inbox link "New" to "This Just In".
    - Change the workflow state name "New" to "Just In".
  - In lead_detail.jsp
    - Change the workflow state name "New" to "Just In".
  - In $BV1TO1/common_erm/alerts/leads/BV_ALERT_TYPES.*xxx*.dmp, where *xxx* is ora, syb, or inf, depending on your database:
    - Change the SQL query from "New" to "Just In".

    See "Using workflow to manage leads" on page 179 for information on how to make your lead active.
- Change the state "Declined" to "Rejected":
  - In lead_inbox.jsp:
    - Change the state name "Declined" to "Rejected".
  - In lead_detail.jsp:
    - Change the state name "Declined" to "Rejected".
- Add a "Hold" state between "Just In" and "Assigned":
  - Add the following lines to lead_detail.jsp:

```
...
var holdID = accessMgr.getWorkflowStateID(service, setID,
    "Hold");
...
form.appendOption(states, holdID, "Hold");
...
```

  - Add code to lead_detail.jsp to take the lead out of alert_spec if moved to this state.
  - In lead_inbox.jsp:
    - Add the line:

```
var holdID = accessMgr.getWorkflowStateID(service, setID,
    "Hold");
```

    - Add "Hold" as a canned inbox option

- Remove the existing workflow path from `"Just In"` to `"Killed"` in the Publishing Center:
  - In `lead_detail.jsp`,remove the ability to transition from `"Just In"` to `"Killed"`.
  - Edit the "Next States" list so the "Just In" workflow state in the Publishing Center does not include "Killed."

7 InfoExchange Portal Admin Tool Component Interfaces

This chapter describes the JavaScript and Java component interfaces used by the InfoExchange Portal Admin Tool.

The JavaScript component interface files are located in `$BV1TO1/include/jsi/bv/webapps`; the Java classes reside in the `com.broadvision.ieportal.components` Java package.

The following component interfaces are used by the InfoExchange Portal Admin Tool.

Interface	Functionality	Page
BVI_KMAdminManager	Implements methods used by the Admin Tool for administering InfoExchange Portal.	187
BVI_EPDistAdminManager	Implements methods used for distributed channel and user administration.	220
BVI_KMAdminChannel	Interfaces with the underlying data type that supports channels.	228
BVI_KMAdminProgram	Interfaces with the underlying data type that supports programs.	234
BVI_KMProgramType	Enables the administration of program types.	246

BVI_KMAdminManager

The BVI_KMAdminManager class implements methods used by the Admin Tool as the starting point for administering InfoExchange Portal.

This component does not allow dynamic properties.

The following section contains information about BVI_KMAdminManager methods.

BVI_KMAdminManager methods

The following table lists the BVI_KMAdminManager methods used by the InfoExchange Portal Admin Tool.

KMAdminManager methods	Functionality
creator()	Creates a new BVI_KMAdminManager object.
creator()	Creates a copy of the specified BVI_KMAdminManager object.
clone()	Creates a copy of this BVI_KMAdminManager object.
get_user_in_any_service()	Returns a list of users. It does a case-insensitive search and will return any user in any service.
getCharacterEncoding()	Gets the character encoding scheme used by the One-To-One Enterprise server.
set_cmc_user_tool_access()	Grants Publishing Center access to a user account.
get_user_list()	Performs a case-insensitive search of the list of visitor accounts.
get_non_site_service_user_list()	Performs a case-insensitive search of the list of visitor accounts in other services. This method is similar to get_user_list(), but excludes any users with site or service admin privileges.
get_user_list_for_user_admin()	Performs a case-insensitive search of the visitors in the service. When given a user_ID, it will return all the users than admin can administer.
getAllAccessGroups()	Returns all Publishing Center access groups for a service.
getGroupsForUser()	Retrieves a list of the access groups to which the visitor has been granted access.
setAccessGroups()	Assigns the visitor to the specified access groups.
parentStatus()	Determines whether the parent channel is On-Line or Off-Line.
leafCategory()	Determines whether the category is a leaf category.
Channel methods	
getChannelListById()	Given a channel ID, returns a list of subchannels.
getAdminChannelListById()	Returns a list of immediate descendent subchannels to the specified channel ID.
getAdminChannelListByName()	Performs a case-insensitive search by name of all channels in the service.
getChannelListByNameForChannelAdminEx()	Returns a list of channels that meet the search criteria, and that are channels the user can administer.
getChannelListByNameForUser()	Given a string, retrieves the specified subchannels for the visitor; given a complete path to a channel, returns the subchannels for the visitor.
getChannelListByNameForUser()	Given a channel name, returns a list of subchannels.
getChannelListByNameEx()	Runs a case-insensitive query and returns all channels that contains the value of the text contained in channelName.
getChannel()	Gets the channel information for the specified channel.

KMAdminManager methods	Functionality
addChannel()	Adds a channel.
updateChannel()	Updates the specified channel.
deleteChannel()	Deletes the specified channel.
Program methods	
getProgramList()	Given a channel ID, returns a list of programs under that channel.
getAdminProgramList()	Returns a list of immediate descendent subchannels to the specified channel ID.
getProgramListByName()	Given a channel name, returns a list of programs under that channel.
getAdminProgramListByName()	Returns programs that contain the search criteria string.
getProgramDisplayOrder()	Returns the display order for the specified program.
getMaxDisplayOrder()	Returns the next available display order of any programs underneath the specified channel.
getProgramParents()	Retrieves the parent channels of the specified program.
getProgram()	Gets a program.
addProgram()	Adds a program to the specified channel.
updateProgram()	Updates the specified program.
deleteProgram()	Deletes the specified program.
Program type methods	
getProgramTypeList()	Gets a list of program types.
isValidProgramType()	Verifies whether the specified program type is valid (not used).
getProgramType()	Gets the program type.
addProgramType()	Adds a program type.
updateProgramType()	Updates a program type.
deleteProgramType()	Removes a program type.
User template methods	
getUserTypeList()	Gets a list of the available user templates.
getUserType()	Gets the user template for the user template ID specified.
addUserType()	Adds a user template.
updateUserType()	Updates a user template.
deleteUserType()	Removes a user template.
Bookmark methods	
setDefaultBookMarkedChannelList()	Sets the default bookmarked channel list by user template ID.
setDefaultBookMarkedProgramList()	Sets the default bookmarked program list by user template ID.
setDefaultBookMarkedContentList()	Sets the default bookmarked content list by user template ID.
getDefaultBookMarkedChannelList()	Gets the default bookmarked channel list by user template ID.
getDefaultBookMarkedProgramList()	Gets the default bookmarked program list by user template ID.

KMAdminManager methods	Functionality
getDefaultBookMarkedContentList()	Gets the default bookmarked content list by user template ID.
Administration methods	
flushContent()	Flushes content by OID.
getUserAdminRights()	Retrieves the visitor's administrative privileges.

BVI_KMAdminManager::creator()

Creates a new BVI_KMAdminManager object.

JavaScript
```
BVI_KMAdminManager creator()
```

Java
```
public BVI_KMAdminManager()
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters None.

Return value A BVI_KMAdminManager object, or null if an error occurs.

BVI_KMAdminManager::creator()

Creates a copy of the specified BVI_KMAdminManager object.

JavaScript
```
BVI_KMAdminManager creator(BVI_KMAdminManager ref)
```

Java
```
public BVI_KMAdminManager(BVI_KMAdminManager ref)
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

`ref`	An existing BVI_KMAdminManager object.

Return value A BVI_KMAdminManager object, or null if an error occurs.

BVI_KMAdminManager::clone()

Creates a copy of this BVI_KMAdminManager object.

JavaScript
```
BVI_KMAdminManager clone()
```

Java `public final BVI_KMAdminManager clone_J()`

Java exception Throws BVWComponentException.

Parameters None.

Return value A copy of the BVI_KMAdminManager object, or null if an error occurs.

BVI_KMAdminManager::getCharacterEncoding()

Gets the character encoding scheme used by the One-To-One Enterprise server.

JavaScript
```
string getCharacterEncoding()
```

Java
```
public final String getCharacterEncoding()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value The character encoding scheme used by One-To-One Enterprise, such as 8859_1, SJIS, or EVCJIS.

BVI_KMAdminManager::set_cmc_user_tool_access()

Grants Publishing Center access to a user account; updates the BV_CP_USER table.

JavaScript
```
void set_cmc_user_tool_access(
    long userId,
    long access);
```

Java
```
public final void set_cmc_user_tool_access(
    long userId,
    long access)
```

Java exception Throws BVWComponentException.

Parameters

`userId`	An integer containing the ID for the current user account.
`access`	An integer determining whether the visitor has Publishing Center access: 0 = no access, 1 = access.

Return value None.

Remarks A value of “0” for “access” denies Publishing Center access; a value of “1” grants it.

BVI_KMAdminManager::get_user_list()

Performs a case-insensitive search for user accounts with criteria that match either the user alias (BV_USER.USER_ALIAS) or user profile name (BV_USER_PROFILE.NAME), and are in the service specified by the service_id parameter.

JavaScript

```
BVI_StringList get_user_list(
    string criteria,
    long service_id
    boolean prefix_query,
    long max_count)
```

Java

```
public final BVI_StringList get_user_list(
    string criteria,
    long service_id
    boolean prefix_query,
    long max_count)
```

Java exception

Throws BVWComponentException.

Parameters

criteria	The search string.
service_id	A string containing the service ID.
prefix_query	Boolean; if true, this method performs a prefix query on the search criteria; if false, no prefix query is performed.
max_count	An integer containing the maximum number of visitor account names to retrieve.

Return value

A BVI_ValueList in which each entry is a BVI_Properties component with the following keys:

USER_ID	User Id
USER_ALIAS	User Alias
NAME	User Profile Name
STATUS	Has a value of “0” if the user account is active; “1” if inactive.

BVI_KMAdminManager::get_non_site_service_user_list()

Performs a case-insensitive search for user accounts with criteria that match either the user alias (BV_USER.USER_ALIAS) or user profile name (BV_USER_PROFILE.NAME). The API will exclude users who are site or service administrators.

JavaScript

```
BVI_ValueList get_non_site_service_user_list(
    string criteria,
    long service_id,
    boolean prefix_query
    long max_count);
```

Java
```
public final BVI_ValueList get_non_site_service_user_list(
    String criteria,
    long service_id,
    boolean prefix_query
    long max_count);
```

Java exception Throws BVWComponentException.

Parameters

criteria	The search string.
service_id	Integer; the ID of the current service.
prefix_query	Boolean; if true, this method performs a prefix query on the search criteria; if false, no prefix query is performed.
max_count	Integer; the maximum number of entries to return.

Return value A BVI_ValueList containing the list. Each entry is a BVI_Properties component with the following keys:

USER_ID	User Id
USER_ALIAS	User Alias
NAME	User Profile Name

Remarks This method does not return user accounts of site or service administrators.

BVI_KMAdminManager::get_user_list_for_user_admin()

Performs a case-insensitive search for user accounts with criteria that matches either the user alias (BV_USER.USER_ALIAS) or user profile name (BV_USER_PROFILE.NAME) Given a USER_ID (user admin ID), it will return all users the administrator can administer.

JavaScript
```
BVI_ValueList get_user_list_for_user_admin(
    string criteria
    string service_id
    string user_id,
    boolean prefix_query,
    long max_count);
```

Java
```
public final BVI_ValueList get_user_list_for_user_admin(
    String criteria,
    String serviceId,
    String user_id,
    boolean prefix_query,
    long max_count);
```

Java exception Throws BVWComponentException.

Parameters

criteria	The search string.
service_id	A string containing the service ID.
user_id	A string containing the user ID (user admin ID).
prefix_query	Boolean; if true, this method performs a prefix query on the search criteria; if false, no prefix query is performed.
max_count	An integer containing the maximum number of visitor account names to retrieve.

Return value

A BVI_ValueList in which each entry is a BVI_Properties component with the following keys:

USER_ID	User Id
USER_ALIAS	User Alias
NAME	User Profile Name
STATUS	Has a value of “0” if the user account is active; “1” if inactive.

Remarks

- Each InfoExchange Portal service is associated with a One-To-One Procurement account. A visitor is assigned to an account, and are thus indirectly added to a service. These accounts are stored in the table MR_ACCT_PROFILE. Visitors assigned to an account have an entry in BV_MR_USER_ACCT.
- This method is intended to be used by User Administrators when they query for visitors in a service.
- If prefix_query is false, the method performs an unconstrained query, which can be slow. To improve the InfoExchange Portal site’s performance, set prefix_query to “true”.

BVI_KMAdminManager::get_user_in_any_service()

Performs a case-insensitive search for user accounts with criteria that match either the user alias (BV_USER.USER_ALIAS) or user profile (BV_USER_PROFILE.NAME). This method performs a search for any user in any service.

JavaScript

```
BVI_ValueList get_user_in_any_service(
    string criteria
    boolean prefix_query,
    long max_count);
```

Java

```
public final BVI_ValueList get_user_in_any_service(
    string criteria
    boolean prefix_query,
    long max_count);
```

Java exception

Throws BVWComponentException.

Parameters

`criteria`	The search string.
`prefix_query`	Boolean; if true, this method performs a prefix query on the search criteria; if false, no prefix query is performed.
`max_count`	An integer containing the maximum number of visitor account names to retrieve.

Return value A BVI_ValueList containing of user accounts that match the search criteria.

BVI_KMAdminManager::getAllAccessGroups()

Gets a list of all access groups for the specified service.

JavaScript
```
BVI_ValueList getAllAccessGroups( string serviceId );
```

Java
```
public final BVI_ValueList getAllAccessGroups(String serviceId)
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the ID for the current service.

Return value A BVI_ValueList in which each entry is a BVI_Property object in which the fields ACCESS_GRP_OID and ACCESS_GRP_NAME are set.

BVI_KMAdminManager::getGroupsForUser()

Retrieves a list of the access groups to which the visitor has been granted access.

JavaScript
```
BVI_ValueList getGroupsForUser( string serviceId,
                                long userId);
```

Java
```
public final BVI_ValueList getGroupsForUser(String serviceId, long
userId)
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`userId`	An integer containing the ID for the current user account.

Return value A BVI_ValueList in which each entry is a BVI_Value object containing the Publishing Center access group ID.

BVI_KMAdminManager::setAccessGroups()

Assigns the visitor to the specified access groups.

JavaScript

```
void setAccessGroups(
    string serviceId
    long userId
    BVI_ValueList valueList
);
```

Java

```
public final void setAccessGroups(
    String serviceId
    long userId
    BVI_ValueList valueList
);
```

Java exception — Throws BVWComponentException.

Parameters

serviceId	A string containing the ID for the current service.
userId	An integer containing the user ID to whom the access groups apply.
valueList	A BVI_ValueList containing a list of OIDs of Publishing Center access groups. Note that a BVI_ValueList contains BVI_Value objects. Each BVI_Value contains an OID(long) that is a Publishing Center access group OID.

Return value — None.

Example

```
var kmMgr = new BVI_KMAdminManager;
var accessGroups = kmMgr.setAccessGroups(serviceId, userId,
accessGroups);
```

BVI_KMAdminManager::parentStatus()

Determines whether the parent channel is On-Line or Off-Line.

JavaScript

```
long parentStatus(
    string serviceId,
    long parentCategoryOID);
```

Java

```
public final long parentStatus(
    String serviceId,
    long parentCategoryOID)
```

Java exception — Throws BVWComponentException.

Parameters

`serviceId`	A string containing the ID for the current service.
`parentCategoryOID`	An integer containing the OID for the category.

Return value If parent is On-Line, returns “1;” if Off-Line, returns“0”

BVI_KMAdminManager::leafCategory()

JavaScript Determines whether a category is a leaf category.

```
long leafCategory( string serviceId,
                   long categoryOID);
```

Java

```
public final long leafCategory(String serviceId, long categoryOID)
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`categoryOID`	An integer containing the OID for the category.

Return value If category is leaf, returns “1;” if non-leaf, returns“0.”

BVI_KMAdminManager::getChannelListById()

Gets a list of subchannels for the specified channel ID.

JavaScript

```
BVI_ValueList getChannelListById(
    string serviceId,
    string channelId,
    long activeFlag );
```

Java

```
public final BVI_ValueList getChannelListById(
    String serviceId,,
    String channelId,
    long activeFlag );
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`channelId`	A string containing the ID for the current channel.
`activeFlag`	An integer; whether the channel is active.

Return value — A BVI_ValueList in which each entry contains a BVI_Properties object. Each of these BVI_Properties objects contains a CHANNEL_OID, CHANNEL_NAME, FULL_PATH (path to channel), STATUS, and PARENT_OID.

BVI_KMAdminManager::getAdminChannelListById()

Returns a list of immediate descendent subchannels to the specified channel ID.

JavaScript

```
BVI_ValueList getChannelListById(
    long serviceId,
    long channelId,
    long userId,
    long isUserAdmin,
    long activeFlag );
```

Java

```
public final BVI_ValueList getChannelListById(
    long serviceId,
    long channelId,
    long userId,
    long isUserAdmin,
    long activeFlag );
```

Java exception — Throws BVWComponentException.

Parameters

serviceId	An integer containing the service ID.
channelId	An integer containing the ID for the current channel.
userId	An integer containing the user ID.
isUserAdmin	An integer; set to “1” if the visitor is a is site or service administrator, otherwise set to “0”.
activeFlag	An integer containing one of the following values: “-2” if the channel is deleted “-1” if the channel can be toggled On-line or Off-line “0” if the channel is Off-line “1” if the channel is On-line

Return value — A BVI_ValueList in which each entry is a BVI_Properties object with the following properties:

CHANNEL_ID	Channel ID.
CHANNEL_NAME	Channel name.
STATUS	Status of the channel; “1” for On-line, “0” for Off-line.
PARENT_ID	Parent ID of the channel.
IS_EDITABLE	Determines whether this channel is editable by this user (the user_id you passed in); “1” for editable, “0” for non-editable.

BVI_KMAdminManager::getAdminChannelListByName()

Performs a case-insensitive search by name of all channels in the service and returns a list of the channels.

JavaScript

```
BVI_ValueList getAdminChannelListByName(
    long serviceId,
    string channelName,
    long userId,
    long isUserAdmin,
    long activeFlag,
    long max_count );
```

Java

```
public final BVI_ValueList getAdminChannelListByName(
    long serviceId,
    String channelName,
    long userId,
    long isUserAdmin,
    long activeFlag,
    long max_count );
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
channelName	A string; the name of the channel being retrieved.
userId	An integer containing the user ID.
isUserAdmin	An integer; set to “1” if the visitor is a is site or service administrator, otherwise set to “0”.
activeFlag	An integer; -1 to display all channels, 0 to display only off-line channels, and 1 to display only on-line channels.
max_count	An integer containing the maximum number of channels to retrieve.

Return value

A BVI_ValueList in which each entry is a BVI_Properties object with the following properties:

CHANNEL_ID	Channel ID.
CHANNEL_NAME	Channel name.
STATUS	Status of the channel; “1” for On-line, “0” for Off-line.
PARENT_ID	ID of the parent channel.
IS_EDITABLE	Determines whether this channel is editable by this visitor; “1” for editable, “0” for non-editable.

BVI_KMAdminManager::getChannelListByName()

Given a string, retrieves the specified subchannels; given a complete path to a channel, returns its subchannels.

JavaScript

```
BVI_ValueList getChannelListByName(
    string serviceId,
    string channelName,
    long activeFlag,
    long max_count );
```

Java

```
public final BVI_ValueList getChannelListByName(
    String serviceId,
    String channelName,
    long activeFlag,
    long max_count );
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`channelName`	A string; the name of the channel being retrieved.
`activeFlag`	An integer; -1 to display all channels, 0 to display only off-line channels, and 1 to display only on-line channels.
`max_count`	An integer containing the maximum number of channels to retrieve.

Return value

A BVI_ValueList in which each entry contains a BVI_Properties object. Each of these BVI_Properties objects contains a CHANNEL_OID, CHANNEL_NAME, FULL_PATH (path to channel), STATUS, and PARENT_OID.

BVI_KMAdminManager::getChannelListByNameForUser()

Given a string, retrieves the specified subchannels for the visitor; given a complete path to a channel, returns the subchannels for the visitor.

JavaScript

```
BVI_ValueList getChannelListByNameForUser(
    string serviceId,
    long userId,
    string channelName,
    long activeFlag,
    long max_count );
```

Java

```
public final BVI_ValueList getChannelListByNameForUser(
    String serviceId,
    long userId,
    String channelName,
    long activeFlag,
    long max_count );
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`userId`	An integer containing the user ID.
`channelName`	A string; the name of the channel being retrieved.
`activeFlag`	An integer; -1 to display all channels, 0 to display only off-line channels, and 1 to display only on-line channels.
`max_count`	An integer containing the maximum number of channels to retrieve.

Return value A BVI_ValueList in which each entry contains a BVI_Properties object. Each of these BVI_Properties objects contains a CHANNEL_OID, CHANNEL_NAME, FULL_PATH (path to channel), STATUS, and PARENT_OID.

BVI_KMAdminManager::getChannelListByNameEx()

Runs a case-insensitive query and returns all channels that contains the value of the text contained the specified search string. Given a `channelExclusionId`, it excludes the channel and its subchannels from the result list.

JavaScript

```
BVI_ValueList getChannelListByNameEx(
    string serviceId,
    string channelName,
    long activeFlag,
    long max_count,
    long channelExclusionId);
```

Java

```
public final BVI_ValueList getChannelListByNameEx(
    String serviceId,
    String channelName,
    long activeFlag,
    long max_count,
    long channelExclusionId);
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`channelName`	A string containing the name of the current channel.
`activeFlag`	An integer; whether the channel is active.
`max_count`	An integer containing the maximum number of channels to retrieve.
`channelExclusionId`	An integer ; whether to exclude the channel and its subchannels from the result list.

Return value A BVI_ValueList in which each entry contains a BVI_Properties object. Each of these BVI_Properties objects contains a CHANNEL_OID, CHANNEL_NAME, FULL_PATH (path to channel), STATUS, and PARENT_OID.

BVI_KMAdminManager::getChannelListByNameForChannelAdminEx()

Returns a list of channels that meet the search criteria and are channels this visitor can administer. This query is run by channel administrators.

JavaScript

```
BVI_KMAdminChannel getChannelListByNameForChannelAdminEx(
    long serviceId,
    string channelName,
    long userId,
    long max_count,
    long channelExclusionId);
```

Java

```
public final BVI_KMAdminChannel getChannelListByNameForChannelAdminEx(
    long serviceId,
    String channelName,
    long userId,
    long max_count,
    long channelExclusionId);
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`channelName`	A string containing the name of the current channel.
`userID`	An integer containing the user ID.
`max_count`	An integer containing the maximum number of channels to retrieve.
`channelExclusionId`	An integer; given a channelExclusionId it will exclude the channel and its subchannels from the result list.

Return value A BVI_ValueList in which each entry contains a BVI_Properties object. Each of these BVI_Properties objects contains a CHANNEL_OID, CHANNEL_NAME, FULL_PATH (path to channel), STATUS (1 on-line, 0 off-line), and PARENT_OID.

BVI_KMAdminManager::getChannel()

Gets the channel information for the specified channel.

JavaScript

```
BVI_KMAdminChannel getChannel( string serviceId, string channelId );
```

Java

```
public final BVI_KMAdminChannel getChannel(String serviceId, String
channelId)
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`channelId`	A string containing the ID for the current channel.

Return value A BVI_KMAdminChannel object containing channel information for `channelId` where `channelId` is the ID of the channel or full channel path name.

BVI_KMAdminManager::addChannel()

Adds a channel.

JavaScript

```
long addChannel(
    string serviceId,
    long userId,
    BVI_KMAdminChannel channel );
```

Java

```
public final long addChannel(
    String serviceId,
    long userId,
    BVI_KMAdminChannel channel );
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`userId`	An integer containing the ID for the current user account.
`channel`	The channel being added.

Return value On success, returns the newly created channel ID; on failure, returns an error code.

Remarks The BVI_KMAdminChannel field parentId contains parent channel where you want to add this channel.

BVI_KMAdminManager::updateChannel()

Updates the specified channel.

JavaScript

```
void updateChannel(
    string serviceId,
    long userId,
    BVI_KMAdminChannel channel,
    long originalChannelParentId,
    long recursive_flag );
```

Java

```
public final void updateChannel(
    String serviceId,
    long userId,
    BVI_KMAdminChannel channel,
    long originalChannelParentId,
    long recursive_flag );
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
userId	An integer containing the ID for the current user account.
channel	A channel containing the new settings.
originalChannelParentId	An integer containing the ID of the original parent channel.
recursive_flag	An integer; defines whether turning a channel on-line or off-line also turns its subchannels on-line or off-line

Return value None.

BVI_KMAdminManager::deleteChannel()

Deletes the specified channel.

JavaScript

```
void deleteChannel(
    string serviceId,
    long userId,
    string channelId );
```

Java

```
public final void deleteChannel(
    String serviceId,
    long userId,
    String channelId );
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
userId	An integer containing the ID for the current user account.
channelId	A string containing the ID of the channel being deleted.

Return value None.

BVI_KMAdminManager::getProgramList()

Retrieves a list of programs.

JavaScript

```
BVI_ValueList getProgramList(
    string serviceId,
    long userId,
    string channelId );
```

Java

```
public final BVI_ValueList getProgramList(
String serviceId,
    long userId,
    String channelId );
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`userId`	An integer containing the ID for the current user account.
`channelId`	A string containing the ID of the current channel.

Return value A BVI_ValueList in which each entry is a BVI_KMAdminProgram object.

BVI_KMAdminManager::getAdminProgramList()

Returns a list of immediate descendent subchannels to the specified channel ID.

JavaScript

```
BVI_ValueList getAdminProgramList(
    long serviceId,
    long userId,
    long isUserAdmin,
    long channelId );
```

Java

```
public final BVI_ValueList getAdminProgramList(
    long serviceId,
    long userId,
    long isUserAdmin,
    long channelId );
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	An integer containing the service ID.
`userId`	An integer containing the user ID.
`isUserAdmin`	An integer; set to “1” if the visitor is a site or service administrator, otherwise set to “0”.
`channelId`	An integer containing the ID for the current channel.

Return value — A BVI_ValueList in which each entry is a BVI_Properties object with the following properties:

PROGRAM_NAME	Program name.
PROGRAM_ID	Program ID.
STATUS	Status of the program; “1” for On-line, “0” for Off-line.
IS_EDITABLE	Determines whether this program is editable by this visitor; “1” for editable, “0” for non-editable.

BVI_KMAdminManager::getProgramListByName()

Retrieves a list of programs having a name containing the string represented by programName.

JavaScript

```
BVI_ValueList getProgramListByName(
    string serviceId,
    string programName,
    long max_count );
```

Java

```
public final BVI_ValueList getProgramListByName(
    String serviceId,
    String programName,
    long max_count );
```

Java exception — Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
programName	A string containing the name of the program.
max_count	An integer containing the maximum number of visitor account names to retrieve.

Return value — A BVI_ValueList in which each entry is a BVI_KMAdminProgram object.

Remarks — The find is case-insensitive.

BVI_KMAdminManager::getAdminProgramListByName()

Returns the programs that contain the search criteria.

JavaScript

```
BVI_ValueList getAdminProgramListByName(
    long serviceId,
    long userId,
    long isUserAdmin,
    string programName,
    long max_count );
```

Java

```
public final BVI_ValueList getAdminProgramListByName(
    long serviceId,
    long userId,
    long isUserAdmin,
    String programName,
    long max_count );
```

Java exception Throws BVWComponentException.

Parameters

serviceId	An integer containing the service ID.
userId	An integer containing the user ID.
isUserAdmin	An integer; set to “1” if the visitor is a site or service administrator, otherwise set to “0”.
programName	A string containing the name of the program.
max_count	An integer containing the maximum number of visitor account names to retrieve.

Return value A BVI_ValueList in which each entry is a BVI_Properties object with the following properties:

PROGRAM_NAME	Program name.
PROGRAM_ID	Program ID.
STATUS	Status of the program; “1” for On-line, “0” for Off-line.
IS_EDITABLE	Determines whether this program is editable by this visitor; “1” for editable, “0” for non-editable.

BVI_KMAdminManager::getProgramDisplayOrder()

Retrieves the display order of a program for the specified channel.

```
long getProgramDisplayOrder(
    string serviceId,
    long channelId,
    string programId );
```

Java

```
public final long getProgramDisplayOrder(
    String serviceId,
    long channelId,
    String programId );
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
channelId	An integer containing the ID of the channel containing the program.
programId	A string containing the ID of the program.

Return value An integer containing the display order of the program for the specified channel ID.

Remarks A program can exist in many channels. In each channel the program can have a different display order. For example, in Channel1 the program Program1 could have a display order of 5, while in Channel2 Program1 could have a display order of 2.

BVI_KMAdminManager::getMaxDisplayOrder()

Gets the maximum display order for the specified channel.

JavaScript

```
long getMaxDisplayOrder(
    string serviceId,
    long channelId );
```

Java

```
public final long getMaxDisplayOrder(
    String serviceId,
    long channelId );
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
channelId	A string containing the ID of the current channel.

Return value An integer containing the maximum display order for a channel.

BVI_KMAdminManager::getProgramParents()

Retrieves the parent channels for the specified program.

JavaScript

```
BVI_ValueList getProgramParents(
    string serviceId,
    string programId);
```

Java

```
public final BVI_ValueList getProgramParents(
    String serviceId,
    String programId);
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
programId	A string containing the ID of the program.

Return value A BVI_ValueList in which each entry contains a BVI_Properties object containing the fields CHANNEL_OID and CHANNEL_PATH.

BVI_KMAdminManager::getProgram()

Retrieves information about a program based on the program ID.

JavaScript

```
BVI_KMAdminProgram getProgram( string serviceId,
                               string programId );
```

Java

```
public final BVI_KMAdminProgram getProgram( String serviceId,
                                            String programId );
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`programId`	A string containing the IDor name of the current program.

Return value

A BVI_KMAdminProgram object containing the specified program.

BVI_KMAdminManager::addProgram()

Adds a program to the specified channel.

JavaScript

```
long addProgram(
    string serviceId,
    long userId,
    long channelId,
    BVI_StringList valueList,
    BVI_KMAdminProgram program );
```

Java

```
public final long addProgram(
    String serviceId,
    long userId,
    long channelId,
    BVI_StringList valueList,
    BVI_KMAdminProgram program)
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`userId`	An integer containing the ID for the current user account.
`channelId`	A string containing the ID of the current channel.
`valueList`	A list of parent channels.
`program`	The program being added.

Return value

The newly-created program OID.

Remarks The BVI_StringList valueList contains a list of parent channels. The first entry is the channel where the program is created. All other entries in valueList reference the newly-created program.

BVI_KMAdminManager::updateProgram()

Updates the specified program.

JavaScript

```
void updateProgram(
    string serviceId,
    long userId,
    long channelId,
    BVI_StringList valueList,
    BVI_KMAdminProgram program );
```

Java

```
public final void updateProgram(
    String serviceId,
    long userId,
    long channelId,
    BVI_StringList valueList,
    BVI_KMAdminProgram program );
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`userId`	An integer containing the ID for the current user account.
`channelId`	A string containing the ID of the current channel.
`valueList`	A list of parent channels.
`program`	The program being updated.

Return value None.

Remarks The BVI_StringList `valueList` contains a list of parent channels. The first entry is the channel where the program is created. All other entries in `valueList` reference the newly-created program.

BVI_KMAdminManager::deleteProgram()

Deletes the specified program from the site.

JavaScript

```
void deleteProgram(
    string serviceId,
    long userId,
    string programId );
```

Java

```
public final void deleteProgram(
    String serviceId,
    long userId,
    String programId );
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`userId`	An integer containing the ID for the current user account.
`programId`	A string containing the ID of the program being deleted.

Return value None.

BVI_KMAdminManager::getProgramTypeList()

Gets a list of all program types for a service.

JavaScript
```
BVI_ValueList getProgramTypeList( string serviceId );
```

Java
```
public final BVI_ValueList getProgramTypeList(String serviceId)
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.

Return value A BVI_ValueList in which each entry contains a BVI_KMProgramType object.

BVI_KMAdminManager::isValidProgramType()

Verifies whether the specified program type exists in this service.

JavaScript
```
boolean isValidProgramType(
    long serviceId,
    long programTypeId );
```

Java
```
public final boolean isValidProgramType(
    long serviceId,
    long programTypeId );
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	A string containing the service ID.
`programTypeId`	A string containing the ID of the program type.

Return value True if the program type is valid; otherwise false.

Remarks This method is not used.

BVI_KMAdminManager::getProgramType()

Retrieves a program type object based on the program type ID.

JavaScript

```
BVI_KMProgramType getProgramType(
    string serviceId,
    string programTypeId );
```

Java

```
public final BVI_KMProgramType getProgramType(
    String serviceId,
    String programTypeId );
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
programTypeId	A string containing the ID of the program type.

Return value

A BVI_KMProgramType object containing information about the program type.

BVI_KMAdminManager::addProgramType()

Adds a program type.

JavaScript

```
long addProgramType(
    string serviceId,
    long userId,
    BVI_KMProgramType programType );
```

Java

```
public final long addProgramType(
    String serviceId,
    long userId,
    BVI_KMProgramType programType );
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
userId	An integer containing the ID for the current user account.
programType	A BVI_KMProgramType structure containing the program type settings.

Return value

An integer containing the new program type ID.

BVI_KMAdminManager::updateProgramType()

Updates the specified program type.

JavaScript

```
void updateProgramType(
    string serviceId,
    long userId,
    BVI_KMProgramType programType );
```

Java

```
public final void updateProgramType(
    String serviceId,
    long userId,
    BVI_KMProgramType programType );
```

Java exception — Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
userId	An integer containing the ID for the current user account.
programType	A BVI_KMProgramType structure containing the program type settings.

Return value — None.

BVI_KMAdminManager::deleteProgramType()

Deletes the specified program type from the system.

JavaScript

```
void deleteProgramType(
    string serviceId,
    long userId,
    string programTypeId );
```

Java

```
public final void deleteProgramType(
    String serviceId,
    long userId,
    String programTypeId );
```

Java exception — Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
userId	An integer containing the ID for the current user account.
programTypeId	A string containing the program type being deleted.

Return value — None.

BVI_KMAdminManager::getUserTypeList()

Gets a list of the available user templates.

JavaScript
```
BVI_ValueList getUserTypeList( string serviceId );
```

Java
```
public final BVI_ValueList getUserTypeList(String serviceId)
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.

Return value A BVI_ValueList in which each object is a BVI_KMUserType object.

BVI_KMAdminManager::getUserType()

Gets the user template for the user template ID specified.

JavaScript
```
BVI_KMUserType getUserType(
    string serviceId
    string userTypeId
);
```

Java
```
public final BVI_KMUserType getUserType(
    String serviceId
    String userTypeId
);
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the ID for the current service.
userTypeId	A string containing the user template ID.

Return value A BVI_KMUserType object, or null if an error occurs.

Example
```
var kmMgr = new BVI_KMAdminManager;
var userType = kmMgr.getUserType(serviceId, userTypeId);
```

BVI_KMAdminManager::addUserType()

Adds a user template.

JavaScript

```
long addUserType(
    string serviceId,
    long userId,
    BVI_KMUserType userType );
```

Java

```
public final long addUserType(
    String serviceId,
    long userId,
    BVI_KMUserType userType );
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the ID for the current service.
userId	An integer containing the ID for the current user account.
userType	The user template being added.

Return value On success, an integer containing the OID of the newly created user template; otherwise reports an error.

BVI_KMAdminManager::updateUserType()

Updates a user template.

JavaScript

```
void updateUserType(
    string serviceId,
    long userId,
    BVI_KMUserType userType );
```

Java

```
public final void updateUserType(
    String serviceId,
    long userId,
    BVI_KMUserType userType );
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the ID for the current service.
userId	An integer containing the ID for the current user account.
userType	The user template being updated.

Return value None.

BVI_KMAdminManager::deleteUserType()

Removes a user template.

JavaScript
```
void deleteUserType(
    string serviceId,
    long userId,
    string userTypeId );
```

Java
```
public final void deleteUserType(
    String serviceId,
    long userId,
    String userTypeId );
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the ID for the current service.
userId	An integer containing the ID for the current user account.
userType	A string containing the ID of the user template being deleted.

Return value None.

BVI_KMAdminManager::setDefaultBookMarkedChannelList()

Sets the default bookmarked channel list for a user template ID.

JavaScript
```
void setDefaultBookMarkedChannelList(
    string serviceId,
    string userTypeId,
    BVI_StringList valueList );
```

Java
```
public final void setDefaultBookMarkedChannelList(
    String serviceId,
    String userTypeId,
    BVI_StringList valueList );
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the ID for the current service.
userTypeId	A string containing the user template ID.
valueList	A BVI_StringList containing the OID for the channels to add to the default bookmarked channel list for the user template.

Return value None.

BVI_KMAdminManager::setDefaultBookMarkedProgramList()

Sets the default bookmarked program list by user template ID.

JavaScript

```
void setDefaultBookMarkedProgramList(
    string serviceId,
    string userTypeId,
    BVI_StringList valueList );
```

Java

```
public final void setDefaultBookMarkedProgramList(
    String serviceId,
    String userTypeId,
    BVI_StringList valueList );
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string containing the ID for the current service.
`userTypeId`	A string containing the user template ID.
`valueList`	A BVI_StringList containing the program OIDs to be added to the default bookmarked program list for the user template.

Return value

None.

BVI_KMAdminManager::setDefaultBookMarkedContentList()

Sets the default bookmarked content list by user template ID.

JavaScript

```
void setDefaultBookMarkedContentList(
    string serviceId,
    string userTypeId,
    BVI_StringList valueList );
```

Java

```
public final void setDefaultBookMarkedContentList(
    String serviceId,
    String userTypeId,
    BVI_StringList valueList );
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	A string containing the ID for the current service.
`userTypeId`	A string containing the user template ID.
`valueList`	A BVI_StringList containing an encoded representing the content items to be added to the default content list for the user template. Each entry contains a service ID number and a content ID number.

Return value

None.

BVI_KMAdminManager::getDefaultBookMarkedChannelList()

Gets the default bookmarked channel list by user template ID.

JavaScript

```
BVI_ValueList getDefaultBookmarkedChannelList(
    string serviceId,
    string userTypeId );
```

Java

```
public final BVI_ValueList getDefaultBookmarkedChannelList(
    String serviceId,
    String userTypeId );
```

Java exception — Throws BVWComponentException.

Parameters

serviceId	A string containing the ID for the current service.
userTypeId	A string containing the user template ID.

Return value — A BVI_ValueList in which each entry contains a BVI_KMChannel object.

BVI_KMAdminManager::getDefaultBookMarkedProgramList()

Gets the default bookmarked program list by user template ID.

JavaScript

```
BVI_ValueList getDefaultBookmarkedProgramList(
    string serviceId,
    string userTypeId );
```

Java

```
public final BVI_ValueList getDefaultBookmarkedProgramList(
    String serviceId,
    String userTypeId );
```

Java exception — Throws BVWComponentException.

Parameters

serviceId	A string containing the ID for the current service.
userTypeId	A string containing the user template ID.

Return value — A BVI_ValueList in which each entry contains a BVI_KMProgram object.

BVI_KMAdminManager::getDefaultBookMarkedContentList()

Gets the default bookmarked content list by user template ID.

JavaScript
```
BVI_ValueList getDefaultBookmarkedContentList(
    string serviceId,
    string userTypeId );
```

Java
```
public final BVI_ValueList getDefaultBookmarkedContentList(
    String serviceId,
    String userTypeId );
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the ID for the current service.
userTypeId	A string containing the user template ID.

Return value A BVI_ValueList in which each entry contains a BVI_KMProperties object with the OID, CONTENT_TYPE, CONTENT_NAME, and SERVICE_ID fields set.

BVI_KMAdminManager::flushContent()

Flushes a content item based on the OID.

JavaScript
```
void flushContent(
    string serviceId,
    long contentType,
    long contentOid);
```

Java
```
public final void flushContent(
    String serviceId,
    long contentType,
    long contentOid);
```

Java exception Throws BVWComponentException.

Parameters

serviceId	A string containing the service ID.
contentType	An integer containing the ID for the content type.
contentOid	An integer containing the OID of the content.

Return value None.

BVI_KMAdminManager::getUserAdminRights()

Retrieves the visitor's administrative privileges.

JavaScript
```
BVI_Properties getUserAdminRights(
    long serviceId,
    long userId);
```

Java

```
public final BVI_Properties getUserAdminRights(
    long serviceId,
    long userId);
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	An integer containing the service ID.
userId	An integer containing the ID for the visitor.

Return value

None.

Remarks

A BVI_Properties object containing the following elements:

IS_SITE_ADMIN	A value of “1” indicates this visitor is a Site Admin, a value of “0” indicates otherwise, the visitor is not of the current service.
IS_SERVICE_ADMIN	A value of “1” indicates this visitor is a Service Admin, a value of “0” indicates otherwise.
IS_CHANNEL_ADMIN	A value of “1” indicates this visitor is a Channel Admin, a value of “0” indicates otherwise.
IS_USER_ADMIN	A value of “1” indicates this visitor is a User Admin, a value of “0” indicates otherwise.

BVI_EPDistAdminManager

The BVI_EPDistAdminManager class defines the methods used for distributed channel and user administration.

This component does not allow dynamic properties.

See the following sections for information about:

- BVI_EPDistAdminManager attributes
- BVI_EPDistAdminManager methods

BVI_EPDistAdminManager attributes

The following table lists the attributes for BVI_EPDistAdminManager.

Attribute	Functionality
EP_DIST_CHANNEL	An integer containing the valid channel node type value(read only).
EP_DIST_USER	An integer containing the valid user account node type value (read only).

BVI_EPDistAdminManager::EP_DIST_CHANNEL

An integer containing the valid channel node type value.

JavaScript
```
readonly attribute long EP_DIST_CHANNEL;
```

Java
```
public final long EP_DIST_CHANNEL();
```

Java exception Throws BVWComponentException.

BVI_EPDistAdminManager::EP_DIST_USER

An integer containing the valid user account node type value.

JavaScript
```
readonly attribute long EP_DIST_USER;
```

ava
```
public final long EP_DIST_USER();
```

Java exception Throws BVWComponentException.

BVI_EPDistAdminManager methods

The following table lists the BVI_EPDistAdminManager methods used by the InfoExchange Portal Admin Tool.

Method	Functionality
creator()	Creates a new BVI_EPDistAdminManager object.
creator()	Creates a copy of the specified BVI_EPDistAdminManager object.
clone()	Creates a copy of this BVI_EPDistAdminManager object (JavaScript).
clone_J()	Creates a copy of this BVI_EPDistAdminManager object (Java).
getAdminsForNode()	Gets a list of administrators for a node.
getParentAdminsForNode()	Gets a list of inherited administrators for a node.
addAdminsForNode()	Adds administrators for a node.
updateAdminsForNode()	Updates administrators for a node.
updateNodesForAdmin()	Update nodes for an administrator.
deleteOrganizationAdmins()	Deletes the administrators for an organization and its suborganizations from the database.
isNodeAdmin()	Verifies whether this visitor is an administrator of this node.
getTopNodesForUser()	Retrieves the top level nodes the visitor can administer.

BVI_EPDistAdminManager::creator()

Creates a new BVI_EPDistAdminManager object.

JavaScript
```
BVI_EPDistAdminManager creator();
```

Java
```
public BVI_EPDistAdminManager creator();
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters None.

Return value A BVI_EPDistAdminManager object, or null if an error occurs.

BVI_EPDistAdminManager::creator()

Creates a copy of the specified BVI_EPDistAdminManager object.

JavaScript
```
BVI_EPDistAdminManager creator(BVI_EPDistAdminManager ref);
```

Java
```
public BVI_EPDistAdminManager creator(BVI_EPDistAdminManager ref);
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

`ref`	An existing BVI_EPDistAdminManager object.

Return value A BVI_EPDistAdminManager object, or null if an error occurs.

BVI_EPDistAdminManager::clone()

Creates a copy of this BVI_EPDistAdminManager object.

JavaScript
```
BVI_EPDistAdminManager clone()
```

Parameters None.

Return value A copy of the BVI_EPDistAdminManager object, or null if an error occurs.

BVI_EPDistAdminManager::clone_J()

Creates a copy of this BVI_EPDistAdminManager object.

Java

```
public final BVI_EPDistAdminManager clone_J()
```

Java exception

Throws BVWComponentException.

Parameters

None.

Return value

A copy of the BVI_EPDistAdminManager object, or null if an error occurs.

BVI_EPDistAdminManager::getAdminsForNode()

Queries the BV_EP_UPROF_DISTAD table and returns a list of administrators for a node. The list excludes the inherited administrators who implicitly have access on the node.

JavaScript

```
BVI_ValueList getAdminsForNode(
    long serviceId,
    long nodeId,
    long nodeType);
```

Java

```
public final BVI_ValueList getAdminsForNode(
    long serviceId,
    long nodeId,
    long nodeType);
```

Java exception

Throws BVWComponentException.

Parameters

`serviceId`	An integer; service ID.
`nodeId`	An integer; node ID.
`nodeType`	An integer: A value of BVI_EPDistAdminManager.EP_DIST_CHANNEL causes the method to update the channel hierarchy, while a value of BVI_EPDistAdminManager.EP_DIST_USER causes the method to update the user account hierarchy.

Return value

Returns a BVI_ValueList in which each element is a BVI_Properties structure containing the following:

`USER_ID`	User ID
`USER_ALIAS`	Login name.
`NAME`	User profile name.

The list is sorted by `NAME`, which is the visitor's BV_USER.USER_ALIAS (login name). Returns null on error.

BVI_EPDistAdminManager::getParentAdminsForNode()

Queries the BV_EP_UPROF_DISTAD table and returns a list of inherited administrators for a node. The list excludes the non-inherited administrators.

JavaScript

```
BVI_ValueList getParentAdminsForNode(
    long serviceId,
    long nodeId,
    long nodeType);
```

Java

```
public final BVI_ValueList getParentAdminsForNode(
    long serviceId,
    long nodeId,
    long nodeType);
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	An integer; service ID
nodeId	An integer; node ID
nodeType	An integer: A value of BVI_EPDistAdminManager.EP_DIST_CHANNEL causes the method to update the channel hierarchy, while a value of BVI_EPDistAdminManager.EP_DIST_USER causes the method to update the user account hierarchy.

Return value

Returns a BVI_ValueList in which each element is a BVI_Properties structure containing the following:

USER_ID	User ID.
USER_ALIAS	Login name.
NAME	User profile name.

The list is sorted by NAME, which is the visitor's BV_USER.USER_ALIAS (login name). Returns null on error.

BVI_EPDistAdminManager::addAdminsForNode()

Adds administrators for a node.

JavaScript

```
void addAdminsForNode(
    long serviceId,
    long nodeId,
    long nodeType,
    BVI_StringList userIdList);
```

Java
```
public final void addAdminsForNode(
    long serviceId,
    long nodeId,
    long nodeType,
    BVI_StringList userIdList);
```

Java exception Throws BVWComponentException.

Parameters

serviceId	An integer; service ID.
nodeId	An integer; node ID.
nodeType	An integer: A value of BVI_EPDistAdminManager.EP_DIST_CHANNEL causes the method to update the channel hierarchy, while a value of BVI_EPDistAdminManager.EP_DIST_USER causes the method to update the user account hierarchy.
userIdList	A BVI_StringList containing the USER_IDs of the new administrators. If the administrator wants to submit an empty list, you need to pass in an empty BVI_StringList component.

Return value None.

BVI_EPDistAdminManager::updateAdminsForNode()

Updates administrators for a node.

JavaScript
```
void updateAdminsForNode(
    long serviceId,
    long nodeId,
    long newParentNodeId,
    long origParentNodeId,
    long nodeType,
    BVI_StringList userIdList);
```

Java
```
public final void updateAdminsForNode(
    long serviceId,
    long nodeId,
    long newParentNodeId,
    long origParentNodeId,
    long nodeType,
    BVI_StringList userIdList);
```

Java exception Throws BVWComponentException.

Parameters

serviceId	An integer; service ID.
nodeId	An integer; node ID.
newParentNodeId	An integer; node ID of new parent node.

`origParentNodeId`	An integer; node ID of original parent node.
`nodeType`	An integer: A value of BVI_EPDistAdminManager.EP_DIST_CHANNEL causes the method to update the channel hierarchy, while a value of BVI_EPDistAdminManager.EP_DIST_USER causes the method to update the user account hierarchy.
`userIdList`	A BVI_StringList containing the USER_IDs of the new administrators. If the administrator wants to submit an empty list, you need to pass in an empty BVI_StringList component.

Remarks If you move a node, pass in the new and original parent node IDs. If you do not move a node, the new and original parent IDs have the same value. This method filters out inherited administrators from the list.

BVI_EPDistAdminManager::updateNodesForAdmin()

Update nodes for an administrator.

JavaScript

```
void updateNodesForAdmin(
    long serviceId,
    long userId,
    long nodeType,
    BVI_StringList nodeIdList);
```

Java

```
public final void updateNodesForAdmin(
    long serviceId,
    long userId,
    long nodeType,
    BVI_StringList nodeIdList);
```

Java exception Throws BVWComponentException.

Parameters

`serviceId`	An integer; service ID.
`userId`	An integer; user ID.
`nodeType`	An integer: A value of BVI_EPDistAdminManager.EP_DIST_CHANNEL causes the method to update the channel hierarchy, while a value of BVI_EPDistAdminManager.EP_DIST_USER causes the method to update the user account hierarchy.
`nodeIdList`	A BVI_StringList containing the category OID of the new nodes. If the administrator wants to submit an empty list, you need to pass in an empty BVI_StringList component.

Return value None.

BVI_EPDistAdminManager::deleteOrganizationAdmins()

Deletes the administrators for an organization and its suborganizations from BV_EP_UPROF_DISTAD table.

JavaScript

```
void deleteOrganizationAdmins(
    long serviceId,
    long nodeId);
```

Java

```
public final void deleteOrganizationAdmins(
    long serviceId,
    long nodeId);
```

Java exception

Throws BVWComponentException.

Parameters

serviceId	An integer; service ID.
nodeId	An integer; node ID.

Return value

None.

BVI_EPDistAdminManager::isNodeAdmin()

Verifies whether this visitor is an administrator of this node.

JavaScript

```
boolean isNodeAdmin(
    long serviceId,
    long userId,
    long nodeId,
    long nodeType);
```

Java

```
public final boolean isNodeAdmin(
    long serviceId,
    long userId,
    long nodeId,
    long nodeType);
```

Parameters

serviceId	An integer; service ID.
userId	An integer; user ID.
nodeId	An integer; node ID.
nodeType	An integer: A value of BVI_EPDistAdminManager.EP_DIST_CHANNEL causes the method to search the channel hierarchy, while a value of BVI_EPDistAdminManager.EP_DIST_USER causes the method to search the user account hierarchy.

Return value Returns “true” if the visitor is an administrator of this node; otherwise returns “false”.

BVI_EPDistAdminManager::getTopNodesForUser()

Retrieves the top level nodes the visitor can administer. Based on the nodeType, it either returns Channel or Organization Entities.

JavaScript

```
BVI_ValueList getTopNodesForUser(
    long serviceId,
    long userId,
    long nodeType);
```

Java

```
public final BVI_ValueList getTopNodesForUser(
    long serviceId,
    long userId,
    long nodeType);
```

Parameters

serviceId	An integer; service ID.
userId	An integer; user ID.
nodeType	An integer: A value of BVI_EPDistAdminManager.EP_DIST_CHANNEL causes the method to retrieve top nodes from the channel hierarchy, while a value of BVI_EPDistAdminManager.EP_DIST_USER causes the method to retrieve them from the user account hierarchy.

Return value A BVI_ValueList in which each entry is a BVI_Properties component with the following name / value pairs:

OID	Node ID.
NAME	Node Name.
STATUS	Node Status; “1” for On-line, “0” for Off-line.
PARENT_OID	Node Parent ID.

BVI_KMAdminChannel

The BVI_KMAdminChannel class interfaces with the underlying data type that supports channels.

This component does not allow dynamic properties.

See the following sections for information about:

- BVI_KMAdminChannel attributes
- BVI_KMAdminChannel methods

BVI_KMAdminChannel attributes

The following table lists the attributes for BVI_KMAdminChannel.

Attribute	Functionality
channelId	An integer containing the channel ID (read only).
serviceId	A string containing the ID for the current service.
parentId	An integer containing the ID of the parent channel.
status	An integer containing the status of the specified channel (0=Off-line and 1=On-line).
channelName	A string containing the name of the specified channel.
channelDescription	A string containing the description of the specified channel.
channelPageType	An integer containing the page type for the specified channel.
channelPagePath	A string containing the path to the start page for the specified channel.
channelMgrName	A string containing the user account name of the manager of the specified channel.
channelMgrEmail	A string containing the e-mail address of the manager of the specified channel.
channelIconPath	A string containing the path to the icon representing the channel.
toggleIconPath	A string containing the path to the fly-over icon for the channel (not used).

BVI_KMAdminChannel::channelId

An integer containing the channel ID.

JavaScript
```
readonly attribute long channelId;
```

Java
```
public final long getChannelId()
```

Java exception Throws BVWComponentException.

BVI_KMAdminChannel::serviceId

A string containing the ID for the current service

JavaScript
```
attribute long serviceId;
```

Java
```
public final long getServiceId()
public final void setServiceId(long serviceId)
```

Java exception Throws BVWComponentException.

BVI_KMAdminChannel::parentId

An integer containing the ID of the parent channel

JavaScript
```
attribute long parentId;
```

Java
```
public final long getParentId()
public final void setParentId(long parentId)
```

Java exception Throws BVWComponentException.

BVI_KMAdminChannel::status

An integer containing the status of the specified channel (0=Off-line and 1=On-line)

JavaScript
```
attribute long status;
```

Java
```
public final long getStatus()
public final void setStatus(long status)
```

Java exception Throws BVWComponentException.

BVI_KMAdminChannel::channelName

A string containing the name of the specified channel.

JavaScript
```
attribute string channelName;
```

Java
```
public final String getChannelName()
public final void setChannelName(String channelName)
```

Java exception Throws BVWComponentException.

BVI_KMAdminChannel::channelDescription

A string containing the description of the specified channel.

JavaScript
```
attribute string channelDescription;
```

Java
```
public final String getChannelDescription()
public final void setChannelDescription(String channelDescription)
```

Java exception Throws BVWComponentException.

BVI_KMAdminChannel::channelPageType

An integer containing the page type for the specified channel.

JavaScript
```
attribute long channelPageType;
```

Java
```
public final long getChannelPageType()
public final void setChannelPageType(long channelPageType)
```

Java exception Throws BVWComponentException.

Remarks A channel can point to either a script or a URL. If a channel points to a script, the value of channelPageType is 0; if a channel points to a URL, the value of channelPageType is 1.

BVI_KMAdminChannel::channelPagePath

A string containing the path to the start page or URL for the specified channel.

JavaScript
```
attribute string channelPagePath;
```

Java
```
public final String getChannelPagePath()
public final void setChannelPagePath(String channelPagePath)
```

Java exception Throws BVWComponentException.

BVI_KMAdminChannel::channelMgrName

A string containing the user account name of the manager of the specified channel.

JavaScript
```
attribute string channelMgrName;
```

Java
```
public final String getChannelMgrName()
public final void setChannelMgrName(String channelMgrName)
```

Java exception Throws BVWComponentException.

BVI_KMAdminChannel::channelMgrEmail

A string containing the e-mail address of the manager of the specified channel.

JavaScript
```
attribute string channelMgrEmail;
```

Java
```
public final String getChannelMgrEmail()
public final void setChannelMgrEmail(String channelMgrEmail)
```

Java exception Throws BVWComponentException.

BVI_KMAdminChannel::channelIconPath

A string containing the path to the icon representing the channel.

JavaScript
```
attribute string channelIconPath;
```

Java
```
public final String getChannelIconPath()
public final void setChannelIconPath(String channelIconPath)
```

Java exception Throws BVWComponentException.

BVI_KMAdminChannel::toggleIconPath

A string containing the path to the fly-over icon for the channel (not used).

JavaScript
```
attribute string toggleIconPath;
```

Java
```
public final String getToggleIconPath()
public final void setToggleIconPath(String toggleIconPath)
```

Java exception Throws BVWComponentException.

BVI_KMAdminChannel methods

The following table lists the methods for BVI_KMAdminChannel.

Method	Functionality
creator()	Creates a new BVI_KMAdminChannel object.
creator()	Creates a copy of the specified BVI_KMAdminChannel object.
clone()	Creates a copy of this BVI_KMAdminChannel object.
setBookmark()	Sets or removes a book mark on a channel for the specified visitor.
isBookmarked()	Verifies whether a channel is bookmarked for the specified visitor.

BVI_KMAdminChannel::creator()

Creates a new BVI_KMAdminChannel object.

JavaScript
```
BVI_KMAdminChannel creator()
```

Java
```
public BVI_KMAdminChannel()
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError.

Parameters None.

Return value A BVI_KMAdminChannel object, or null if an error occurs.

BVI_KMAdminChannel::creator()

Creates a copy of the specified BVI_KMAdminChannel object.

JavaScript
```
BVI_KMAdminChannel creator(BVI_KMAdminChannel ref)
```

Java
```
public BVI_KMAdminChannel(BVI_KMAdminChannel ref)
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError.

Parameters

`ref`	An existing BVI_KMAdminChannel object.

Return value A BVI_KMAdminChannel object, or null if an error occurs.

BVI_KMAdminChannel::clone()

Creates a copy of this BVI_KMAdminChannel object.

JavaScript
```
BVI_KMAdminChannel clone()
```

Java
```
public final BVI_KMAdminChannel clone_J()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value A copy of the BVI_KMAdminChannel object, or null if an error occurs.

BVI_KMAdminChannel::setBookmark()

Sets or removes a book mark on a channel for the specified visitor.

JavaScript
```
void setBookmark( long userId, boolean flag );
```

Java
```
public final void setBookmark(long userId, boolean flag)
```

Java exception Throws BVWComponentException.

Parameters

`userId`	An integer containing the ID number of the visitor account.
`flag`	A boolean; if true, bookmark is set; if false, bookmark is not set.

Return value None.

BVI_KMAdminChannel::isBookmarked()

Verifies whether a channel is bookmarked for the specified visitor.

JavaScript
```
boolean isBookmarked( long userId );
```

Java
```
public final boolean isBookmarked(long userId)
```

Java exception Throws BVWComponentException.

Parameters

`userId`	An integer containing the ID number of the visitor account.

Return value True if the channel is bookmarked for this visitor; false if the channel is not bookmarked for this visitor.

BVI_KMAdminProgram

The `BVI_KMAdminProgram` class interfaces with the underlying data type that supports programs.

This component does not allow dynamic properties.

See the following sections for information about:

- BVI_KMAdminProgram attributes
- BVI_KMAdminProgram methods

BVI_KMAdminProgram attributes

The following table lists the attributes for BVI_KMAdminProgram.

Attribute	Functionality
programId	An integer containing the ID number of the program (read only).
serviceId	An integer containing the service ID (read only).
creationTime	The date and time the program was created (read only).
programName	A string containing the name of the program.
programDescription	A string containing the description of the program.
programPageType	An integer representing the page type.

Attribute	Functionality
programPagePath	A string containing the path to the start page for the program.
status	An integer representing the program status; “0” for Off-Line, “1” for On-Line.
deleted	An integer indicating whether the program has been deleted.
programType	An integer representing the program type.
programMgrName	A string containing the name of the program manager.
programMgrEmail	A string containing the e-mail address of the program manager.
lastModTime	When the program was last modified (read only).
programSrcType	An integer representing the program source type.
programSrcId	An integer representing the program source type.
programSrcContentType	An integer representing the program source content type (read only).
displayOrder	An integer; the order in which the program is displayed.
includeSubcategories	An integer representing whether the program includes subcategories.
showOnlyReadable	An integer; whether to only show programs the visitor can read.
programIconPath	A string containing the path to the icon representing the program.
toggleIconPath	A string representing the path to the fly-over icon representing the program.
embedScriptPath	A string containing the path to an embedded script.
pubScriptPath	A string containing the path to Instant Publisher script for publishing to that program.
pubCategoryId	A string containing the OID of the category to which the Instant Publisher script writes.
intcol1	An integer; can be used for additional fields.
intcol2	An integer; can be used for additional fields.
intcol3	An integer; can be used for additional fields.
strcol1	A string; can be used for additional fields.
strcol2	A string; can be used for additional fields.
strcol3	A string; can be used for additional fields.
primaryParentId	An integer; can be used for additional fields.
cacheable	An integer; determines whether the program is cacheable.
timeout	An integer; the timeout for the cache in seconds.

BVI_KMAdminProgram::programId

An integer containing the ID number of the program (read only).

JavaScript

```
readonly attribute long programId;
```

Java

```
public final long getProgramId()
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::serviceId

An integer containing the service ID (read only).

JavaScript
```
readonly attribute long serviceId;
```

Java
```
public final long getServiceId()
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::creationTime

The date and time the program was created (read only).

JavaScript
```
readonly attribute BVI_DateTime creationTime;
```

Java
```
public final BVI_DateTime getCreationTime()
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::programName

A string containing the name of the program.

JavaScript
```
attribute string programName;
```

Java
```
public final String getProgramName()
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::programDescription

A string containing the description of the program.

JavaScript
```
attribute string programDescription;
```

Java
```
public final String getProgramDescription()
public final void setProgramDescription(String programDescription)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::programPageType

An integer representing the page type.

JavaScript
```
attribute long programPageType;
```

Java
```
public final long getProgramPageType()
public final void setProgramPageType(long programPageType)
```

Java exception Throws BVWComponentException.

Remarks A program can point to either a script or a URL. If a program points to a script, the value of programPageType is 0; if a program points to a URL, the value of programPageType is 1.

BVI_KMAdminProgram::programPagePath

A string containing the path to the start page or URL for the program.

JavaScript
```
attribute string programPagePath;
```

Java
```
public final String getProgramPagePath()
public final void setProgramPagePath(String programPagePath)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::status

An integer representing the program status; “0” for Off-Line, “1” for On-Line.

JavaScript
```
attribute long status;
```

Java
```
public final long getStatus()
public final void setStatus(long status)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::deleted

An integer indicating whether the program has been deleted.

JavaScript
```
attribute long deleted;
```

Java
```
public final long getDeleted()
public final void setDeleted(long deleted)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::programType

An integer representing the program type ID the program uses.

JavaScript
```
attribute long programType;
```

Java
```
public final long getProgramType()
public final void setProgramType(long programType)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::programMgrName

A string containing the name of the program manager.

JavaScript
```
attribute string programMgrName;
```

Java
```
public final String getProgramMgrName()
public final void setProgramMgrName(String programMgrName)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::programMgrEmail

A string containing the e-mail address of the program manager.

JavaScript
```
attribute string programMgrEmail;
```

Java
```
public final String getProgramMgrEmail()
public final void setProgramMgrEmail(String programMgrEmail)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::lastModTime

When the program was last modified (read only).

JavaScript
```
readonly attribute BVI_DateTime lastModTime;
```

Java
```
public final BVI_DateTime getLastModTime()
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::programSrcType

An integer representing the program source type.

JavaScript
```
attribute long programSrcType;
```

Java
```
public final long getProgramSrcType()
public final void setProgramSrcType(long programSrcType)
```

Java exception Throws BVWComponentException.

Remarks A programSrcType value of 0 indicates a rule set, a value of 1 indicates a category, and a value of 2 indicates a content item.

BVI_KMAdminProgram::programSrcId

An integer representing the program source type.

JavaScript
```
attribute long programSrcId;
```

Java
```
public final long getProgramSrcId()
public final void setProgramSrcId(long programSrcId)
```

Java exception Throws BVWComponentException.

Remarks A programSrcId value of 0 indicates a rule set, a value of 1 indicates a category, and a value of 2 indicates a content item.

BVI_KMAdminProgram::programSrcContentType

An integer containing the program source content type ID (read only).

JavaScript
```
readonly attribute long programSrcContentType;
```

Java
```
public final long getProgramSrcContentType()
```

Java exception Throws BVWComponentException.

Remarks Content type is retrieved from the program type.

BVI_KMAdminProgram::displayOrder

An integer; the order in which the program is displayed.

JavaScript
```
attribute long displayOrder;
```

Java
```
public final long getDisplayOrder()
public final void setDisplayOrder(long displayOrder)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::includeSubcategories

An integer representing whether the program includes subcategories.

JavaScript
```
attribute long includeSubcategories;
```

Java
```
public final long getIncludeSubcategories()
public final void setIncludeSubcategories(long includeSubcategories)
```

Java exception Throws BVWComponentException.

Remarks This attribute is only valid if the program points to a category.

BVI_KMAdminProgram::showOnlyReadable

An integer; whether to only show programs the visitor can read.

JavaScript
```
attribute long showOnlyReadable;
```

Java
```
public final long getShowOnlyReadable()
public final void setShowOnlyReadable(long showOnlyReadable)
```

Java exception Throws BVWComponentException.

Remarks If this attribute is set, check Publishing Center access control to verify the visitor has access to the content of this program.

BVI_KMAdminProgram::programIconPath

A string containing the path to the icon representing the program.

JavaScript
```
attribute string programIconPath;
```

Java
```
public final String getProgramIconPath()
public final void setProgramIconPath(String programIconPath)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::toggleIconPath

A string representing the path to the fly-over icon representing the program.

JavaScript
```
attribute string toggleIconPath;
```

Java
```
public final String getToggleIconPath()
public final void setToggleIconPath(String toggleIconPath)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::embedScriptPath

A string containing the path to an embedded script.

JavaScript
```
attribute string embedScriptPath;
```

Java
```
public final String getEmbedScriptPath()
public final void setEmbedScriptPath(String embedScriptPath)
```

Java exception Throws BVWComponentException.

Remarks You can create a program that points to a script. This script can display any data and is visitor-defined. The site administrator can then include this program, or *embedded script* path, inside a block.

BVI_KMAdminProgram::pubScriptPath

A string containing the path to an Instant Publisher form script.

JavaScript
```
attribute string pubScriptPath;
```

Java
```
public final String getPubScriptPath()
public final void setPubScriptPath(String pubScriptPath)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::pubCategoryId

An integer containing the category ID (`PubCategoryId()`). This category is used to hold contents published directly to the program.

JavaScript
```
attribute long pubCategoryId;
```

Java
```
public final attribute long getPubCategoryId()
public final void setPubCategoryId(long pubCategoryId)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::intcol1

An integer; can be used for additional fields.

JavaScript
```
attribute long intcol1;
```

Java
```
public final long getIntcol1()
public final void setIntcol1(long intcol1)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::intcol2

An integer; can be used for additional fields.

JavaScript
```
attribute long intcol2;
```

Java
```
public final long getIntcol2()
public final void setIntcol2(long intcol2)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::intcol3

An integer; can be used for additional fields.

JavaScript
```
attribute long intcol3;
```

Java
```
public final long getIntcol3()
public final void setIntcol3(long intcol3)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::strcol1

A string; can be used for additional fields.

JavaScript
```
attribute string strcol1;
```

Java
```
public final String getStrcol1()
public final void setStrcol1(String strcol1)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::strcol2

A string; can be used for additional fields.

JavaScript
```
attribute string strcol2;
```

Java
```
public final String getStrcol2()
public final void setStrcol2(String strcol2)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::strcol3

A string; can be used for additional fields.

JavaScript
```
attribute string strcol3;
```

Java
```
public final String getStrcol3()
public final void setStrcol3(String strcol3)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::primaryParentId

The ID of the primary parent channel ID. This method is for distributed administration. A program can only belong to one PRIMARY_PARENT_ID. If a user can administer that PRIMARY_PARENT_ID they can administer the program. A PRIMARY_PARENT_ID is a CHANNEL_OID.

JavaScript
```
attribute long primaryParentId;
```

Java
```
public final long getPrimaryParentId()
public final void setPrimaryParentId(long primaryParentId)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::cacheable

Determines whether the program can be cached. A value of 1 mean yes; a value of 0 means no.

JavaScript
```
attribute long cacheable;
```

Java
```
public final long getCacheable()
public final void setCacheable(long cacheable)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram::timeout

The timeout for the program cache, in seconds.

JavaScript
```
attribute long timeout;
```

Java
```
public final long getTimeout()
public final void setTimeout(long timeout)
```

Java exception Throws BVWComponentException.

BVI_KMAdminProgram methods

The following table lists the BVI_KMAdminProgram methods used by the InfoExchange Portal Admin Tool.

Method	Functionality
creator()	Creates a new BVI_KMAdminProgram object.
creator()	Creates a copy of the specified BVI_KMAdminProgram object.
clone()	Creates a copy of this BVI_KMAdminProgram object.
setBookmark()	Bookmarks the program for the specified visitor.
isBookmarked()	Determines whether the specified visitor has bookmarked this program.
getContentList()	Gets a list of content items in a program for the specified visitor.

BVI_KMAdminProgram::creator()

Creates a new BVI_KMAdminProgram object.

JavaScript
```
BVI_KMAdminProgram creator()
```

Java
```
public BVI_KMAdminProgram()
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError.

Parameters None.

Return value A BVI_KMAdminProgram object, or null if an error occurs.

BVI_KMAdminProgram::creator()

Creates a copy of the specified BVI_KMAdminProgram object.

JavaScript
```
BVI_KMAdminProgram creator(BVI_KMAdminProgram ref)
```

Java
```
public BVI_KMAdminProgram(BVI_KMAdminProgram ref)
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

`ref`	An existing BVI_KMAdminProgram object.

Return value A BVI_KMAdminProgram object, or null if an error occurs.

BVI_KMAdminProgram::clone()

Creates a copy of this BVI_KMAdminProgram object.

JavaScript
```
BVI_KMAdminProgram clone()
```

Java
```
public final BVI_KMAdminProgram clone_J()
```

Java exception Throws BVWComponentException.

Parameters None.

Return value A copy of the BVI_KMAdminProgram object, or null if an error occurs.

BVI_KMAdminProgram::setBookmark()

Bookmarks the program for the specified visitor.

JavaScript
```
void setBookmark( long userId, boolean flag );
```

Java
```
public final void setBookmark(long userId, boolean flag)
```

Java exception Throws BVWComponentException.

Parameters

`userId`	An integer containing the ID number of the visitor account.
`flag`	A boolean; if true, bookmark is set; if false, bookmark is not set.

Return value None.

BVI_KMAdminProgram::isBookmarked()

Determines whether the specified visitor has bookmarked this program.

JavaScript

```
boolean isBookmarked( long userId );
```

Java

```
public final boolean isBookmarked(long userId)
```

Java exception

Throws BVWComponentException.

Parameters

`userId`	An integer containing the ID number of the visitor account.

Return value

True is the visitor has bookmarked this program; false if the visitor has not bookmarked this program.

BVI_KMAdminProgram::getContentList()

Gets a list of content items in a program for the specified visitor.

JavaScript

```
BVI_ContentList getContentList(
    long user_id,
    BVI_SessnProf sprof,
    long max_count );
```

Java

```
public final BVI_ContentList getContentList(
    long user_id,
    BVI_SessnProf sprof,
    long max_count );
```

Java exception

Throws BVWComponentException.

Parameters

`userId`	An integer containing the ID number of the visitor account.
`sprof`	The visitor's session profile.
`max_count`	A n integer; the maximum number of content items to be displayed.

Return value

A BVI_ContentList object containing all content items available to the visitor for this program.

BVI_KMProgramType

The BVI_KMProgramType class is used by the Admin Tool to define program types.

This component does not allow dynamic properties.

BVI_KMProgramType attributes

The following table lists the attributes for `BVI_KMProgramType`.

Attribute	Functionality
typeId	An integer; the ID of the program type.
typeName	A string; the name of the program type.
programContentType	An integer; the type of content in the program.
ruleSetCategory	An integer; the rule set category used to define the content of the program type.
typeIconPath	A string; the path from the script-root to the icon representing the program type.
toggleIconPath	Not used.
serviceId	An integer; the ID of the current service.
status	An integer; indicates whether the program type is On-Line or Off-Line.
deleted	An integer; indicates whether the program type has been deleted.

BVI_KMProgramType::typeId

An integer; the ID of the program type.

JavaScript
```
attribute long typeId
```

Java
```
public final long getTypeId()
public final void setTypeId(long typeId)
```

Java exception Throws BVWComponentException.

BVI_KMProgramType::typeName

A string; the name of the program type.

JavaScript
```
attribute string typeName
```

Java
```
public final String getTypeName()
public final void setTypeName(String typeName)
```

Java exception Throws BVWComponentException.

BVI_KMProgramType::programContentType

An integer; the type of content in the program.

JavaScript
```
attribute long programContentType
```

Java
```
public final long getProgramContentType()
public final void setProgramContentType(long programContentType)
```

Java exception Throws BVWComponentException.

BVI_KMProgramType::ruleSetCategory

An integer; the rule set category used to define the content of the program type.

JavaScript
```
attribute long ruleSetCategory
```

Java
```
public final long getRuleSetCategory()
public final void setRuleSetCategory(long ruleSetCategory)
```

Java exception Throws BVWComponentException.

BVI_KMProgramType::typeIconPath

A string; the path from the script-root to the icon representing the program type.

JavaScript
```
attribute string typeIconPath
```

Java
```
public final String getTypeIconPath()
public final void setTypeIconPath(String typeIconPath)
```

Java exception Throws BVWComponentException.

BVI_KMProgramType::toggleIconPath

A string; the path from the script-root to the fly-over icon.

JavaScript
```
attribute string toggleIconPath
```

Java
```
public final String getToggleIconPath()
public final void setToggleIconPath(String toggleIconPath)
```

Java exception Throws BVWComponentException.

BVI_KMProgramType::serviceId

An integer; the ID of the current service.

JavaScript
```
readonly attribute long serviceId
```

Java
```
public final long getServiceId()
```

Java exception Throws BVWComponentException.

BVI_KMProgramType::status

An integer; indicates whether the program type is On-Line or Off-Line.

JavaScript
```
attribute long status
```

Java
```
public final long getStatus()
public final void setStatus(long status)
```

Java exception Throws BVWComponentException.

BVI_KMProgramType::deleted

An integer; indicates whether the program type has been deleted from the One-To-One Enterprise system.

JavaScript
```
attribute long deleted
```

Java
```
public final long getDeleted()
public final void setDeleted(long deleted)
```

Java exception Throws BVWComponentException.

BVI_KMProgramType methods

The following methods are defined by BVI_KMProgramType:

Method	Functionality
creator()	Creates a BVI_KMProgramType object.
creator()	Returns a copy of an existing BVI_KMProgramType object.
clone()	Returns a copy of this project or null on error.

BVI_KMProgramType::creator()

Creates a BVI_KMProgramType object.

JavaScript
```
creator;
```

Java
```
public BVI_KMProgramType()
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError.

Return value An instance of BVI_KMProgramType; otherwise an error flag is raised.

BVI_KMProgramType::creator()

Returns a copy of an existing BVI_KMProgramType object.

JavaScript
```
creator(BVI_KMProgramType ref);
```

Java
```
public BVI_KMProgramType(ref)
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError.

Parameters

`ref`	An existing BVI_KMProgramType object.

Return value A BVI_KMProgramType object; otherwise an error flag is raised.

BVI_KMProgramType::clone()

Returns a copy of this object, or null on error.

JavaScript
```
BVI_KMProgramType clone;
```

Java
```
public final BVI_KMProgramType clone_J()
```

Java exception Throws BVWComponentException.

Return value A copy of this object, or null on error.

Example
```
var prog_type = new BVI_KMProgramType;
var clone1 = prog_type.clone;
```

8 Developing Portlets

A *portlet* encapsulates an information source external to BroadVision in a way that makes that information usable in a visitor's configurable home page or anywhere in the portal navigation hierarchy. Portlets support the display and integration of external data such as a news feed or an HTML or XML fragment pulled from an external web site.

InfoExchange Portal implements a Portlet framework for development, deployment, administration, execution and configuration of Portlets. A library of Portlets is available for InfoExchange Portal, including an HTTP Portlet to incorporate information from any web site and a Reuters Portlet to display news.

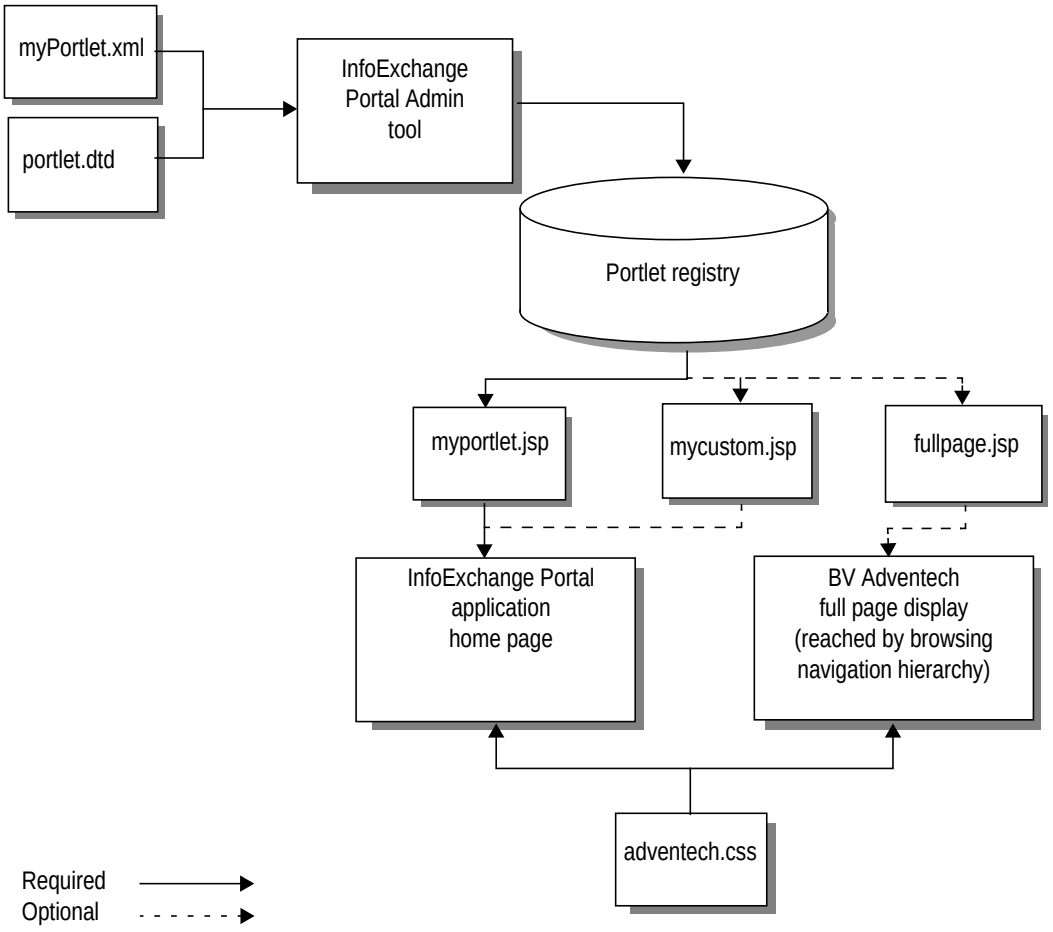
Portlets can be created to access relevant information in the enterprise—whether that information resides in databases, applications, other web sites, or external data feeds, such as stock quotes and news. Any application or information provider can create portlets by developing to the InfoExchange portlet specification.

Portlets are contained in programs; they can be added to blocks and configured on the home page like any other program. A visitor to the page does not need to be aware of the content source for a program; the portlet displays the content transparently. Programs containing portlets can be configured in exactly the same way other programs are configured: an InfoExchange Portal administrator can configure and target each portlet using only the Admin Tool.

The following topics are discussed in this chapter:

Portlet overview

A portlet consists of an XML registration file and one or more *script* files. The following diagram shows how portlets are registered and displayed in an InfoExchange Portal application.



1. An XML registration file (for example, `myPortlet.xml)` defines all of the attributes to be used by the script files for the portlet. This file is written by the portlet developer and must conform to `$BV1TO1/portlets/portlet.dtd`, a document type definition (DTD) provided with InfoExchange Portal.

2. The site administrator uses the Admin Tool to register the portlet. The registration process parses the XML file and populates tables in the BroadVision database that serve as the portlet registry. The site administrator can also set the values of the portlet attributes.

3. The site, service or channel administrator uses the Admin Tool to create a program pointing to the portlet, sets any program-specific portlet attributes, and adds the program to blocks on one or more home pages.

4. Up to three script files display the portlet content.
   a. The required script (for example, `myPortlet.jsp`) is a *block script* that retrieves portlet attributes from the database, passes the attributes into the application, and generates HTML code to display the content. For more information about the block script, see "Creating a block script" on page 259.
   b. (Optional) If the visitor is allowed to edit some of the portlet attribute values, the portlet must contain a *visitor configuration script* (for example, `mycustom.jsp`) that provides a form for editing the attributes and updating the attributes in the database. For more information about visitor configuration scripts, see "Creating a visitor configuration script" on page 260.
   c. (Optional) If the portlet is to be displayed in a different format in full page mode, the portlet must include a *page script* that controls what is shown when a visitor is browsing the navigation hierarchy of the site. The page script can have a different layout than the block script. For more information about page scripts, see "Creating a script for displaying a portlet in full page mode" on page 261.
5. InfoExchange Portal applications should include a cascading style sheet that determines the formatting for the application, and portlet scripts should use styles, not formatting tags.

Portlets available from BroadVision

InfoExchange Portal ships with several fully developed portlets, and additional portlets are available at the BroadVision FTP site.

Portlets on the InfoExchange Portal CD

When you install InfoExchange Portal, the installation program creates the `portlets` directory in the $BV1TO1 directory. The `portlets` directory contains two subdirectories:

- `jsp` for portlets developed in JavaServer Pages.
- `scripts` for portlets developed in JavaScript.

Both subdirectories contain the same set of portlets. Each individual portlet directory includes a readme file in PDF format called *`portlet_name`*`_readme.pdf`. These readme files describe the portlet files and any additional steps needed to make the portlet function on your site.

example The example portlet does not provide any functionality. Its purpose is to serve as a model, showing developers how to structure their code when writing a portlet.

HTTP The HTTP portlet retrieves a piece of any external web page and parses the output so that the links in the external site work in the portlet.

alerts The alerts portlet displays an inbox of messages triggered when content changes or is added to a program for which the visitor has set an alert. It only displays the output of triggered alerts, not alerts that have been set but not triggered.

bookmarks The bookmarks portlet displays a set of links to visitor-selected channels, programs, and content items. It also shows the default bookmarks selected by the administrator for the visitor's user template.

reuters The Reuters portlet retrieves news headlines in HTML from the Reuters News Service. You must be a subscriber of the Reuters News Service to use this portlet.

Portlets on the BroadVision FTP site

More portlets will be available on the BroadVision FTP site, ftp.broadvision.com.

Implementing a portlet

Implementing a portlet for use with an InfoExchange Portal program requires two major steps:

- Write the necessary files and scripts for defining and displaying the portlet.

  This step, described in "Developing a portlet" on page 254, requires one or more developers who are familiar with XML and JavaServer Pages or JavaScript.

- Register the portlet using the InfoExchange Portal Admin Tool.

  This step, described in "Administering portlets" on page 262, requires an administrator who is familiar with the InfoExchange Portal application and its users. Administrators can also use the Admin Tool to modify or delete the portlet.

Developers must also register the portlet in a test application to verify that it works before the administrator registers it with a live site.

Portlet design strategies

To get the most out of a portlet, keep the following in mind as you design it:

- Keep portlet code short.

  The block script in particular should be short because it is run from within a home page that performs several other tasks in addition to calling the portlet. Portlets run serially, so no individual portlet should take a long time to return.

- Use portlets primarily for display-oriented functionality.

  The InfoExchange Portal application does not provide APIs for portlets to refresh pages after submitting form data. Therefore, display-oriented functionality is easier to write, and does not usually require code specific to the application calling the portlet. Customers can write a portlet that uses forms or other interactive functionality, but the development process for such portlets is more difficult and time consuming.

- Plan for reuse.
  - Use portlet attributes instead of hard-coding values like the destinations for links.
  - Use program-scope attributes if a visitor might want several instances of the portlet with different parameters.
  - Use style sheets to format output.

Developing a portlet

Portlet developer tasks include:

- Creating a portlet XML registration file that conforms to the DTD provided with InfoExchange Portal. For more information, see "Writing the portlet XML registration file" on page 255.
- Write one or more portlet scripts using JavaServer Pages or JavaScript. For more information, see "Writing portlet script files" on page 258.

- Running the portlet in a test application, which requires registering the portal with the test application as described in "Registering a portlet" on page 263.

Writing the portlet XML registration file

The registration file specifies values for the standard portlet attributes, such as the suggested name, description, and the location of the other scripts used by the portlet. When an administrator registers the portlet, the XML registration file is read by the Admin Tool. This file must conform to the DTD included with InfoExchange Portal. The registration file format is identical for JavaScript and Java portlets. The file is located in:

```
$BV1TO1/portlets/portlet.dtd
```

The following table describes the tags used in an XML registration file. Note that the element order, tag name case, and attribute value case are all significant.

Tag or attribute	Description
<portlet></portlet>	Root element for the DTD. This element contains the following child elements:
<pName>*text*</pName>	Portlet name; appears in the field "Portlet Name" in the Admin Tool and in the Heading bar on the visitor's home page. **Required**.
<pVersion>*text* </pVersion>	The version of the portlet, for example, 1.0; appears in but cannot be modified in the Admin Tool.
<pDescription>*text* </pDescription>	Text describing the functionality of the portlet.
<pBlockScript>*path* </pBlockScript>	The relative path from the portlet directory to the block script. **Required**.
<pPageScript>*path* </pPageScript>	The relative path from the portlet directory to the page script, if there is one.
<pVisitorScript>*path* </pVisitorScript>	The relative path from the portlet directory to the visitor configuration script, if there is one.
<pType type = "*text*" />	Portlet type; indicates whether the portlet is written in Java or JavaScript. Possible values are "JavaScript" or "JSP". **Required**. Cannot be modified in the Admin Tool.
<pIEPVersion>*N.N.N* </pIEPVersion>	The version of InfoExchange Portal for which this portlet was written; appears in but cannot be modified in the Admin Tool.
<pDynamicAttribute>*list* </pDynamicAttribute>	A list of values describing a single dynamic attribute. There can be multiple aDynamicAttribute elements in an XML registration file. This element contains the following child elements.
<aId>*text*</aId>	The attribute ID. This must be less than ten alphanumeric characters long and must start with an alphabetic character. **Required**.
<aFriendlyName>*text* </aFriendlyName>	Descriptive text that appears in the Admin Tool to the left of the input field.
<aScope scope = "*text*" />	The scope of the attribute. Valid values are "site-wide", "program", or "visitor." The value of this element cannot be changed in the Admin Tool. The default is "site-wide". **Required**.

Tag or attribute	Description
<aControlType control-type = "*text*" />	The input format as it will appear in the Admin Tool. Valid values are "selectbox", "radiobutton","checkbox", "text", or "textarea". **Required**.
<aControlValue>*text* </aControlValue>	If the Control type is "selectbox" or "radiobutton", these are the possible values of the control type.
<aInitialValue>*value* </aInitialValue>	The initial attribute value. This is the value of the attribute as it first appears when registering the portlet in the Admin Tool.
<aRequired required = "*text*" />	Whether this attribute is required; must be either "yes" or "no". The default is "no".

The remainder of this section describes some of the DTD elements and issues in more detail.

pType

The scripts used for displaying portlets on an InfoExchange Portal page must be written in either JavaServer Pages or JavaScript. Which language a developer writes portlets in depends on the following factors:

- If the portlet is to be displayed on a configurable home page, the portlet scripts must be written in the same language as that home page.
- If multiple portlets are to be displayed on a page, the portlet scripts must be written in the same language.

Directory paths for pBlockScript, pVisitorScript, and pPageScript

The directory paths for the various scripts used by a portlet are defined relative to the portlet directory. The portlet directory is relative to the script root, which is defined when the Interaction Manager is configured (for more information, see the *One-To-One Enterprise Installation and System Administration Guide*). By default, it is simply `script-root/portlets`. You can change this location by modifying the startup script `$BV1TO1_VAR/lib/script_library/bv_ep_utils.js` to change the value of the global variable `ep_portlet_script_path`.

The default portlet directory structure includes two directories under the portlet directory called `jsp` and `scripts`, containing JavaServer Page portlets and JavaScript portlets respectively. Portlet developers should follow this convention when specifying the path to their scripts in the XML file.

pDynamicAttribute and aScope

Portlets may require dynamic attributes to be set at any or all of three possible scopes:

Site-wide settings apply to the portlet in general and are set in the portlet configuration pages of the Admin Tool. For a Microsoft Outlook portlet, the name of the Outlook server might be a site-wide parameter; for a news feed portlet, the URL for the news feed might be a site-wide parameter.

Since each portlet is registered in a particular service, the site-wide settings are actually specific to a service.

Program-specific settings may be different, depending on which node in the hierarchy is displaying the portlet. This allows the portlet code to be generic, while allowing the administrator to modify some aspects of the portlet depending upon where it is to be displayed. The general behavior of the

portlet is the same everywhere, but the program-specific portlet settings to call may be different for different programs. The values of these attributes are set in the **Channels and Programs** page of the InfoExchange Portal Admin Tool.

For example, the MS Outlook portlet might not have any program-specific parameters. However, in a news feed portlet, each program could specify a different news category for the feed. In this paradigm you can reuse the same integration code many times, as well as target it by using qualifiers.

Visitor-specific settings are different for each visitor. These are set in the visitor configuration script provided with the portlet. For an MS Outlook portlet, the visitor might have an e-mail address and password. However, a news feed portlet might not have any visitor-editable attributes. In this case, visitors will not see an edit button for that portlet in their configurable home page.

When you add attributes to a portlet, you must also add attribute IDs (aId tag). Attribute IDs must have an alphanumeric name that starts with an alphabetic character. These names must be less than ten characters long.

 Attribute ID names are case-sensitive.

A pDynamicAttribute may have only one value selected at a time. If the portlet needs to allow multiple potential values, the XML file must contain one attribute for each value that can be displayed. For example, if the portlet allows visitors to specify several stock symbols they are interested in, the XML file must include attributes SYM1, SYM2, SYM3, and so on. Only attributes that have values are added to the visitor's profile; this avoids creating empty columns in the profile table.

aControlType and aControlValue

The aControlType element determines the input format for setting the value of a pDynamicAttribute in the Admin Tool. If the control type is a select box or radio button, the attribute has two or more possible values. Each possible value must be defined as an aControlValue element in the XML registration file.

A checkbox has only two values: true (if checked) and false. True is represented numerically as a 1; False is represented numerically as a 0.

If the aControlType element is a select box or radio button, each possible value that can be selected from the select box must be defined as an aControlValue element. The following code from `$BV1TO1/portlets/jsp/example/reg_file_example.xml` shows an example of a select box in a registration file.

```
<pDynamicAttribute>
    <aId>COLOR</aId>
    <aFriendlyName>Color (Example of a select box)</aFriendlyName>
    <aScope scope = "program" />
    <aControlType control-type = "selectbox" />
    <aControlValue>black</aControlValue>
    <aControlValue>blue</aControlValue>
    <aControlValue>teal</aControlValue>
    <aControlValue>red</aControlValue>
    <aControlValue>purple</aControlValue>
    <aInitialValue>black</aInitialValue>
    <aRequired required = "yes" />
</pDynamicAttribute>
```

Text box, text area, and check boxes control types do not require control values. If the registration file defines an aControlValue element for a text box, text area, or checkbox aControlType, those control values are ignored.

Writing portlet script files

In addition to the XML registration file, portlet developers must write at least one JavaServer Page or JavaScript to display the portlet in an InfoExchange Portal application.

- The required script is a *block script* that executes in one block of the home page, retrieves portlet attributes from the database, passes the attributes into the application, and generates HTML code to display the content. For information about creating this script, see "Creating a block script" on page 259.
- (Optional) If the visitor is allowed to edit the attribute values, the portlet must contain a *visitor configuration script* that provides a form for editing the attributes and updating the attributes in the database. For information about creating this script, see "Creating a visitor configuration script" on page 260.
- (Optional) If the portlet is to be displayed in a different format in full page mode, the portlet must include a *page script* that controls what is shown when a visitor is browsing the navigation hierarchy of the site. The page script can have a different layout than the block script. For information about creating this script, see "Creating a script for displaying a portlet in full page mode" on page 261.

The contents of these files may differ depending on whether you are writing the portlet in Java or JavaScript. All of the files written for a specific portal must be in the same language. InfoExchange Portal is shipped with an example portlet implemented in both JavaScript and JavaServer Pages to help demonstrate how to write a portlet. The example portlet files are located in:

JavaScript: `$BV1TO1/portlets/scripts/example`

JavaServer Pages: `$BV1TO1/portlets/jsp/example`

These example scripts contain segments of "boilerplate" code that can be used as-is as well as code that needs to be changed or customized for the portlet being developed. The following sections describe the boilerplate and portlet-specific code for each of the three scripts.

Formatting HTML output

Portlet developers should not hard-code font formatting into their HTML output. Instead, they should use the following CSS styles, which allows the calling application to control the look and feel of the portlet.

- bodyCopy—for most non-link text, especially on the home page.
- bodyTitle—for titles on the home page and anywhere else a small title is necessary.
- contentLink—for hyperlinks that should look more or less like the rest of the text.
- subtitle—for titles that need to be larger than bodyTitle. This style should not be used in block scripts.
- lgLink—for hyperlinks that should stand out from the rest of the text. This style should be used sparingly.

Using BroadVision components in portlet development

To create new portlets, you do not need to know how to build BroadVision components or how to modify the BroadVision schema. However, you can call any of the components available in InfoExchange Portal or One-To-One Enterprise. These components can be called from JavaServer Pages or JavaScript. Examples of components that you can use to write portlets are:

- BVI_HTTPPortal, for making HTTP requests (one use is in the HTTP portlet which exposes this component to the Admin tool and does some post processing on the results).
- BVI_IDLPortal, which allows access from Java Server Pages to CORBA servers without writing new components
- BVI_SessionPortal, which allows access to other BroadVision systems, applications and integrations. Systems can be completely separated, such as a partner's BroadVision system.
- The COM extension, a BroadVision component for Windows NT that allows communication with any COM object—a Windows interface standard by which Microsoft products share information. This is accessible to UNIX systems through BVI_SessionPortal.
- The system portal, which allows you to call any executable on your server.

See the *Application Programmer's Reference* for more information about using these components.

Creating a block script

The block script is the one required script file for a portlet. The block script executes in one block of the configurable home page. The script:

- Retrieves portlet attributes.
- (Optional) Calls BroadVision components or other back-end systems.
- Outputs an HTML fragment.

The boilerplate code does the following:

1. Retrieves necessary attributes from the previous page and the Session and assigns them to variables.

   These attributes include the userID (of the current BVI_Visitor instance), the serviceID for the application, and the ID of the program containing this portlet.

2. Instantiates components needed by every portlet.

   Each block script must instantiate BVI_EPPortletManager and retrieve the portlet itself by running BVI_EPPortletManager::getPortlet().

The script developer is responsible for writing:

- The script code that calls any other components needed by the portlet.

  This code should assume statelessness. If the portlet requires any additional variables from the Session object, the developer should retrieve the needed properties and assign them to variables in the block script code. This code must also verify that the variables are not null before using them because the calling application may not have instantiated them.

In JavaScript block scripts, all variables must begin with `ptlt_` to avoid possible conflicts with variables in the calling page.

- The HTML code that displays the portlet on the home page.
  - This code must be a small fragment that does *not* contain the <HTML> or <BODY> tags (it is safe to assume that the calling page includes these tags).
  - This HTML must not interfere with anything else on the home page. For example, any HTML tables must include all tags for starting and ending the table.

JavaServer Pages implementing portlets must also include code that loads any component libraries needed for the portlet. The examples load BroadVision components necessary for the portlet to work. Portlet developers must add their own libraries as needed.

Creating a visitor configuration script

The visitor configuration script, which is optional, allows the visitor to set the values of any visitor-scope portlet attributes. If there is a visitor configuration script included for the portlet, an **Edit** button appears in the program that displays the portlet on the home page. When the visitor clicks the **Edit** button, the application calls a page that includes the script.

Visitor configuration scripts must be written in the same language as the block script. The example visitor configuration scripts (`visitor_config_example.jsp` in both the *`$BV1TO1`*`/portlets/scripts/example` and *`$BV1TO1`*`/portlets/jsp/example` directories) contain alternating segments of "boilerplate" code and code that needs to be changed or customized for the portlet being developed.

Boilerplate

The first segment of boilerplate code retrieves necessary attributes from the previous page and the Session and assigns them to variables.

These attributes include:

- The userID (of the current BVI_Visitor instance)
- The serviceID for the application
- The ID of the program containing the portlet
- The destination page
- The error page for the visitor configuration page
- Instantiates the portlet.

Customized

In the first segment of code to be changed, portlet developers must customize the code that initializes the values of the attributes the visitor is permitted to modify. In the example, the two attributes the visitor can modify are the text of the greeting (`greeting`) and the number of stars displayed (`number`).

Boilerplate

The second segment of boilerplate code:

- Retrieves the current visitor portlet attributes from the BV_EP_UPROF_PTLT table.
- Manages errors or exceptions thrown in the process of retrieving the attributes.

Customized

The second section of code to be changed requests parameters back from the form and sets values for the portlet attributes. To customize this section, portlet developers must:

- Replace the attributes and values with those from their own portlets.
- Write code to handle errors or exceptions.

When coding in Java, any value entered in a text box is treated as a string and must be explicitly converted to a long, float, or integer if a numeric type is required. It is best practice to cast all values retrieved from an HTML form.

Boilerplate — The third segment of boilerplate updates the visitor's attributes and redirects either to the destination or error page previously defined.

Customized — The third section of code to be changed is the HTML code that appears on the visitor's **Edit** page for the portlet. This code should be a fragment that does *not* contain the <HTML> or <BODY> tags (it is safe to assume that the calling page includes these tags).

Boilerplate — The last section of the visitor configuration page contains hidden input to be passed to the next page. This section must not be changed.

Creating a script for displaying a portlet in full page mode

The page script, which is optional, determines how the portlet is to be displayed by itself on a page. If there is no page script for a portlet, the portlet is displayed on a full page the same way it is displayed on the home page. The only difference between the page script and the block script is that there is more screen space available to a page script, and the page script can format its output for a full page view.

Page scripts must be written in the same language as the block script. The example page scripts (`page_script_example.jsp` in both the `$BV1TO1/portlets/scripts/example` and `$BV1TO1/portlets/jsp/example` directories) contain alternating segments of "boilerplate" code and code that needs to be changed or customized for the portlet being developed.

Boilerplate — The first segment of boilerplate code retrieves necessary attributes from the previous page and the Session and assigns them to variables.

These attributes include:

- The userID (of the current BVI_Visitor instance)
- The serviceID for the application
- The programID for the program containing the portlet

This code also initializes an instance of the BVI_EPPortletManager and retrieves the portlet associated with the page script. It also manages error and exception handling for these tasks.

Customized — The customized part of this script must include:

- Any components needed to make the portlet work, such as a new instance of BVI_HTTPPortal() or a new BVI_KMProgramManager().
- Default values for all variables (in case the site administrator has not entered the values when registering the portal).
- Code that retrieves values from other components as necessary (such as the visitor name).
- The HTML code that displays the portlet on the page.
  - This code should be a fragment that does *not* contain the <HTML> or <BODY> tags (it is safe to assume that the calling page includes these tags).
  - This HTML must not interfere with anything else on the home page. For example, any HTML tables must include all tags for starting and ending each table.

Administering portlets

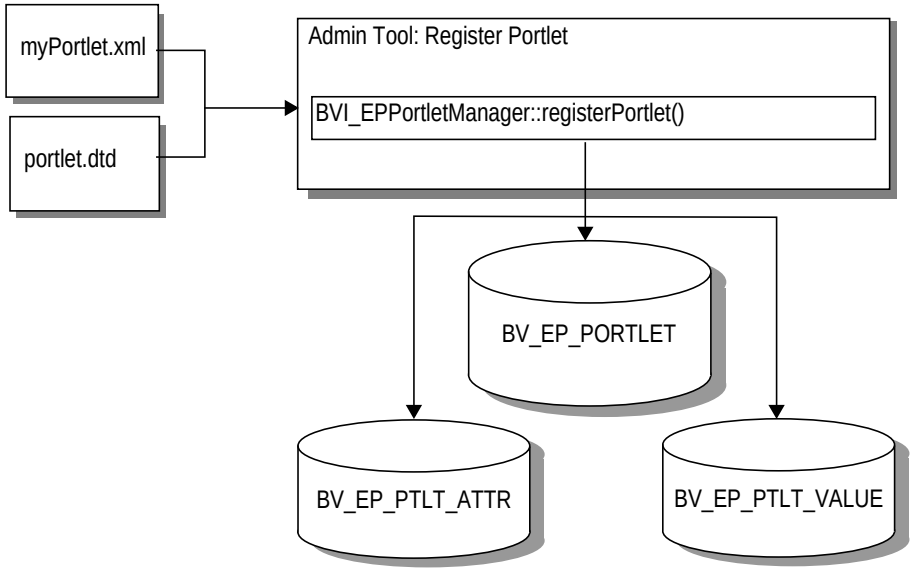
InfoExchange Portal provides a portlet registry in the One-To-One Enterprise database to manage and maintain the administrative tasks. The **Portlets** option in the left menu of the InfoExchange Portal Admin Tool enables you to administer portlets and manage their attributes. Once you have registered a Portlet, you can add it to a program.

This section describes the following topics:

To perform Portlet administration with the Admin Tool, you must have Site Admin or Service Admin privileges.

Portlet registry process

The following diagram shows the major steps and tables used in registering a Portlet.



When an administrator registers a portlet in the Admin Tool, the application calls BVI_EPPortletManager::registerPortlet(). This method:

1. Parses the XML registration file according to `portlet.dtd`.
2. Populates three tables with the data gathered from the registration file:

- BV_EP_PORTLET, which contains the portlet metadata, such as its name and type.
- BV_EP_PTLT_ATTR, which contains the dynamic attributes defined for the portlet.
- BV_EP_PTLT_VALUE, which contains the control values for the portlet's dynamic attributes.

For a description of the information contained in these tables, see "Portlet tables" on page 338 and "Writing the portlet XML registration file" on page 255.

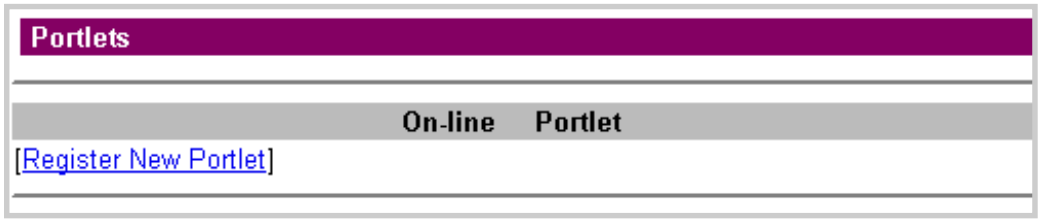
Registering a portlet

Before you register a portlet, you need to copy the files provided by the portlet developer into the *script_root*/portlets directory.

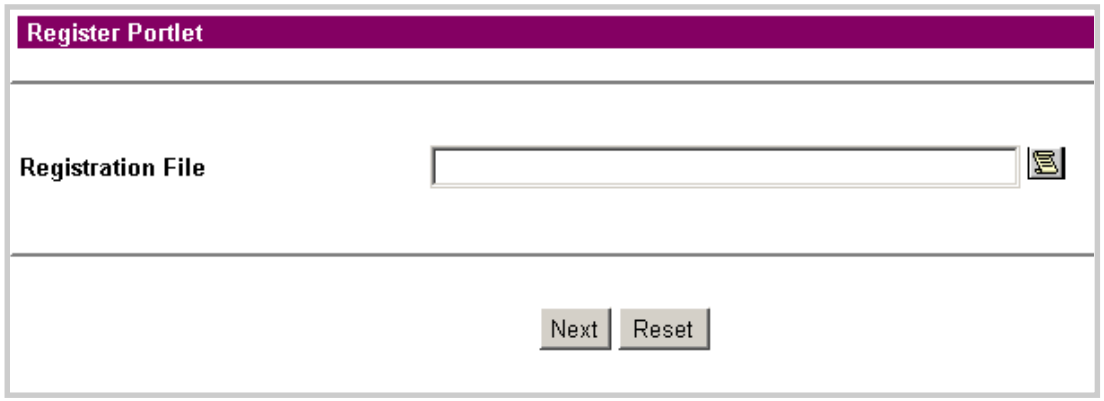
To register a portlet select **Register New Portlet** in the Portlets window.

These are the steps you must follow to register a portlet using the Portlets window:

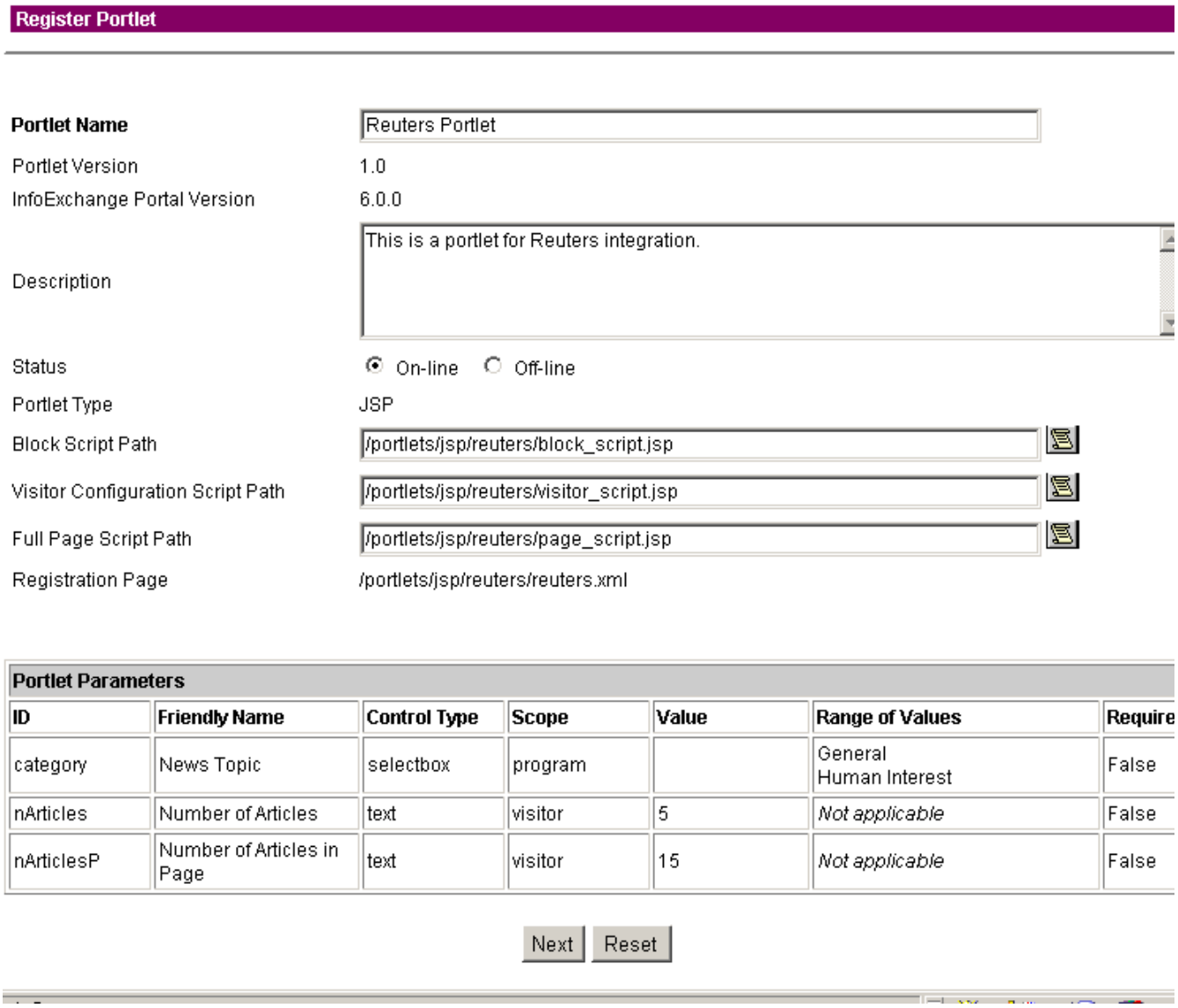
1. Select **Portlets** in the Admin Tool. The Portlets window appears.



2. Select **[Register New Portlet]**. The Register Portlet window appears.



3. Enter the pathname from the `portlets` directory or use the File Dialog icon to select the XML **Registration File**. To revert to the original values from the XML registration file, select the **Reset** button; otherwise, select the **Next** button to display the next page in the registration process.



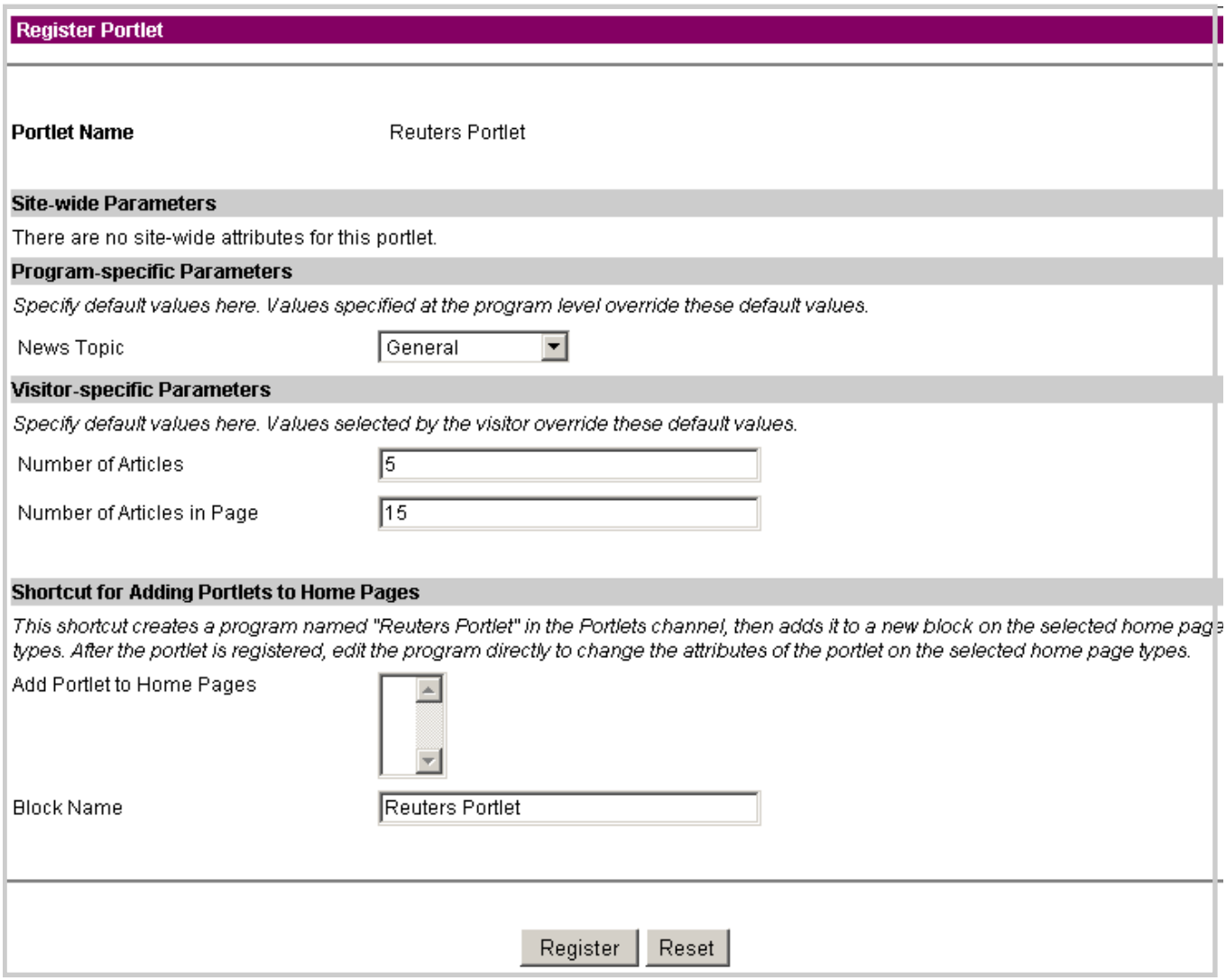
4. Modify portlet attributes in the first page of the registration form.

   The values for the portlet attributes are initially set in the XML registration file, but you can override them by changing them in this page. You can edit the following attribute values in this page:

   - **Portlet Name** - the name of the portlet (required).
   - **Description** - text description of the portlet (optional).
   - **Status** - whether the portlet is **On-line** or **Off-line** (required)
   - **Block Script Path** - the path from the portlets directory to the block script (required)

- **Visitor Configuration Script Path** - the path from the portlets directory to the block script (optional)
- **Full Page Script Path** - the path from the portlets directory to the page script (optional)
- The **Portlet Version**, **InfoExchange Portal Version**, **Portlet Type**, **Registration Page**, and **Portlet Parameters** are displayed but cannot be changed in this window.

To undo any changes you have made to this page of the form, select the **Reset** button; otherwise, select the **Next** button to display the second page of the registration form.



5. In this page, check and enter or modify he default values for the following:
   - **Site-wide Parameters** - parameters that pertain to the use of the portlet in the current service.
   - **Program-specific Parameters** - parameters that are local to a program. These values are defaults. If changed, the values do not propagate to programs already created.
   - **Visitor-specific Parameters** - parameters local to each visitor. These values are defaults. If changed, the values do not propagate to visitors who have set their own values.

   For more information on these parameters, see "pDynamicAttribute and aScope" on page 256.

6. (Optional) Complete the **Shortcut for Adding Portlets to Home Pages**. Here you can:
   - **Add Portlet to Home Pages** - Select home page types to which this portlet is added.
   - **Block Name** - Change the name of the portlet as it appears in the block on the visitor's home page.

If you use this shortcut, the Admin Tool goes through the following steps behind the scenes:

- It creates a program with the same name as the portlet. This program is in the **Portlets** channel (which is off-line by default so visitors cannot navigate to it). This program has no qualifiers set, making it visible to all visitors who select it.
- It creates an optional block on each selected home page with the name of the portlet.
- It adds the newly-created program to this optional block.

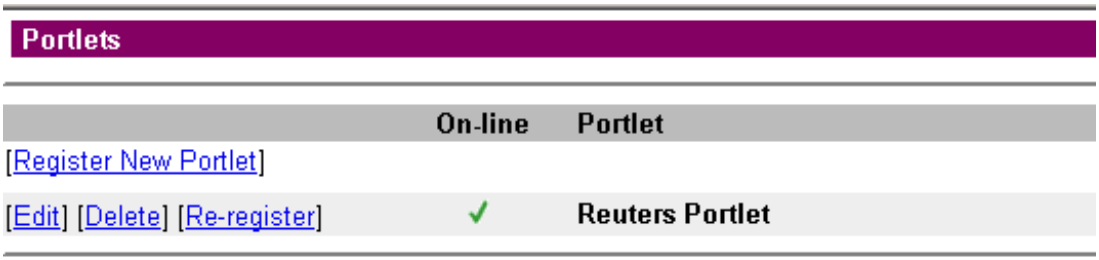
You can modify programs and blocks created with the shortcut just like any that were created explicitly with the InfoExchange Portal Admin Tool. To add a portlet to an existing program, see "Adding a portlet to an existing program" on page 268.

7. To undo your changes, click the **Reset** button; otherwise, click the **Register** button to register the new portlet into a channel named Portlets.

Editing a portlet

These are the steps you must follow to edit a portlet:

1. Select **Portlets** in the Admin Tool. The Portlets window appears.



2. Select **[Edit]** to the left of the name of the portlet you want to edit.

   This page contains most of the same options as the initial registration pages described in Step 4 on page 264 and Step 5 on page 265. It has a different page title (Edit Portlet) and does not include the shortcut step at the bottom of the second initial registration pages.

   In this page you can set or modify the following portlet attribute values:

   - **Portlet Name** - the name of the portlet (required).
   - **Description** - text description of the portlet (optional).
   - **Status** - whether the portlet is **On-line** or **Off-line** (required)
   - **Block Script Path** - the path from the portlets directory to the block script (required)
   - **Visitor Configuration Script Path** - the path from the portlets directory to the block script (optional)
   - **Full Page Script Path** - the path from the portlets directory to the page script (optional)
   - **Site-wide Parameters** - parameters that pertain to the use of the portlet in the current service.

- **Program-specific Parameters** - parameters that are local to a program. These values are defaults. If changed, the values do not propagate to programs already created.
- **Visitor-specific Parameters** - parameters local to each visitor

The **Portlet Version**, **InfoExchange Portal Version**, **Portlet Type**, **Registration Page**, and **Portlet Parameters** are displayed but cannot be changed in this window.

3. To undo your changes, click the **Reset** button; otherwise, click the **Save** button to save the changes you have made to the portlet attributes.

Deleting a portlet

To delete a portlet, in the Portlets window, select **[Delete]** to the left of the name of the portlet you want to delete. A pop-up window appears asking you to confirm your decision with the following warning:

> WARNING:
>
> Deleting a portlet will also delete all programs containing the portlet. If those programs are included in blocks on a visitor's home page, those blocks will be empty. If you intend to register this portlet again, you should choose Re-register, which will preserve the relationship with existing programs, rather than Delete.


Deleting the portlet also deletes the portlet attributes from the visitor profile.

If you decide you do not want to delete the portlet, select the Cancel button; otherwise select the OK button to delete the portlet.

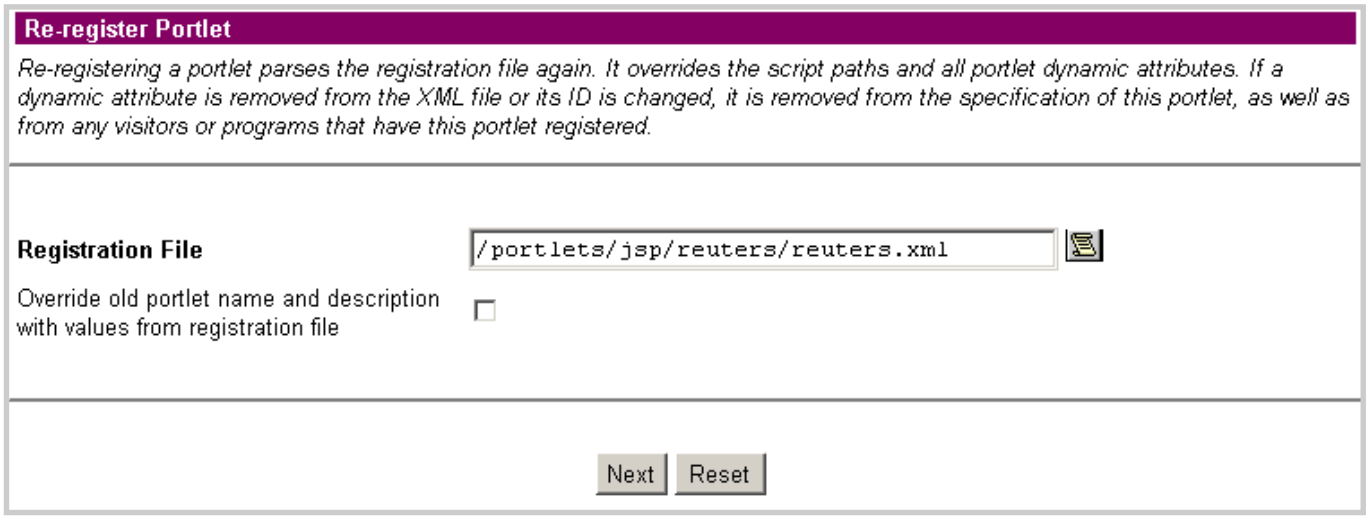
Re-registering a portlet

If you have made significant changes to a portlet that is already assigned to existing programs, you can re-register the portlet. This enables you to add or delete portlet parameters and functionality while keeping the portlet in the program. If a program with a re-registered portlet is included in a block on a visitor's home page, the block will still contain content.


Re-registering a portlet parses the registration file again. It overrides the script paths and all portlet dynamic attributes. If a dynamic attribute is removed from the XML file or its ID is changed, it is removed from the specification of this portlet, as well as from any visitors or programs that have this portlet registered.

To re-register a portlet, in the Portlets window:

1. Select **[Re-register]** to the left of the name of the portlet you want to re-register to display the registration file.



2. By default, the re-registration process overrides only the dynamic attributes (pDynamicAttribute) and script paths from the original XML registration file. To override all registration file parameters, including Name and Description, select the checkbox to the right of **Override old portlet name and description with values from registration file.**

3. Click the Next button to display the first page of the Re-register Portlet form.

   This page contains most of the same options as the initial registration pages described in Step 4 on page 264 and Step 5 on page 265. It has a different page title (Re-register Portlet).

4. Click the Reset button to set the values back to their original values. If you chose a different registration file, then click Reset, it will revert to the values from the original registration file.

Adding a portlet to an existing program

There are reasons you may not want to use the shortcut for registering portlets:

- You want to register the portlet in a channel other than Portlets.
- You want to specify different attributes when the portlet runs under different programs.
- You want to set qualifiers on the portlet.

If you have any of these requirements, follow these steps to add a portlet to a program. For a full description of the steps, see the “Site Presentation” chapter in the *InfoExchange Administrator’s Guide.*

1. Navigate to the channel in which you want to register this portlet.

2. Create a new program or modify an existing one.

3. Add the portlet as the content source for a new or existing program. Set the appropriate program-scope parameters.

4. Edit the page type and add a new block or modify an existing block to point to the program.

BVI_EPPortletManager

The BVI_EPPortletManager class supports the registration and management of portlets. This section contains information about BVI_EPPortletManager methods, including both the methods typically used by portlet developers and the methods used to develop the portlet pages in the Admin Tool. Unless you choose to write your own administration tool, you do not need to use the latter methods.

This component does not allow dynamic properties.

This section describes the following topics:

- "Portlet development methods" on page 269.
- "Portlet administration methods" on page 272.

Portlet development methods

This section describes the methods portlet developers are likely to use.

BVI_EPPortletManager methods	Functionality
creator()	Creates a new BVI_EPPortletManager object.
getPortlet()	Retrieves attributes of the specified portlet as a BVI_Properties object.
getDefault()	Returns the value of a dynamic attribute default or initial value.
getAttrsByScope():	Gets attributes by scope for a portlet in name/value pairs.
getAttrsForVisitor()	Gets visitor-specific attributes for the specified portlet for a visitor.
setAttrsForVisitor()	Sets one or more visitor specific dynamic attributes in a portlet for a visitor.

BVI_EPPortletManager::creator()

Creates a new BVI_EPPortletManager object. It is important to create a Portlet Manager before using any Portlet functions.

JavaScript

```
BVI_EPPortletManager creator( long serviceId, long visitorId )
```

Java

```
public BVI_EPPortletManager creator( long serviceId, long visitorId )
```

Java exception

Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

serviceId	A long containing the service ID
visitorId	A long containing the ID for the current visitor

Return value A BVI_EPPortletManager object, or null if an error occurs.

BVI_EPPortletManager::getPortlet()

Retrieves attributes of the specified portlet as a BVI_Properties object.

```
BVI_Properties getPortlet(
    long portletId,
    long visitorId,
    long programId);
```

Parameters

`portletId`	OID for the portlet.
`visitorId`	Visitor ID.
`programId`	Program ID.

Return value A BVI_Properties object that contains the following:

- PTLT_NAME portlet name
- PTLT_DESC portlet description
- PTLT_VERSION portlet version
- PTLT_BLOCK_SCPT portlet block script path
- PTLT_PAGE_SCPT portlet page script path
- PTLT_VIST_SCPT portlet visitor script path
- PTLT_TYPE portlet type
- PTLT_REG_FILE portlet registration file path
- PTLT_IEP_VER version of IEP used to develop the portlet.

It also includes name / value pairs for the dynamic attributes specified in the XML file. Values for the attributes are retrieved from different tables depending on the scope of the attribute. If the attribute scope is “visitor,” it looks first in the visitor table (the ATTR_VALUE column of BV_EP_UPROF_PTLT). If no visitor ID is provided or the specified visitor does not have a value for that attribute, it returns the default value from the portlet table (ATTR_INITIAL from BV_EP_PORTLET). Similarly, for attributes that have “program scope” the method returns first the value in the program table (ATTR_VALUE column of the BV_EP_PROG_PTLT) or the default value from the portlet table if there is no program specified or the specified program does not have a value for the attribute.

Remarks Both `visitorId` and `programId` can have the value 0. However, If the value of `portletId` is 0, the value of `programId` must not be 0 because it is needed for retrieving `portletId`. If visitorId is 0, the visitor table is ignored and the portlet’s default values are used for visitor scope attributes. Either portletId or programId must be specified (non-zero). If programId is specified, program-scope attributes are retrieved from the program; otherwise, the default values for the portlet are used.

BVI_EPPortletManager::getDefault()

Returns the value of a dynamic attribute default or initial value.

```
string getDefault
    (string attrName,
     long portletId);
```

Parameters

attrName	Name of the attribute.
portletId	OID for the specified portlet.

BVI_EPPortletManager::getAttrsByScope():

Gets attributes by scope for a portlet in name / value pairs.

```
BVI_Properties getAttrsByScope(
    long portletId,
    long scope);
```

Parameters

portletId	OID for the portlet.
scope	Attribute scope. The value for site-wide is 0, for program-specific is 1, and for visitor-specific is 2.

Return value — Attributes for the specified portlet as attribute-ID / value pairs for all dynamic attributes by “scope.”

BVI_EPPortletManager::getAttrsForVisitor()

Get visitor-specific attributes for the specified portlet for a visitor.

```
BVI_Properties getAttrsForVisitor(
    long portletId,
    long visitorId,
    long programId);
```

Parameters

portletId	OID for the portlet.
visitorID	Visitor ID.
programId	Program ID.

Return value — ATTR_ID / ATTR_VALUE pairs for a portlet by “visitor.”

Remarks — Visitors have their own values for visitor-specific dynamic attributes. This function is used to retrieve these values.

Both `visitorId` and `programId` can have the value 0. If `visitorId` is 0, the visitor table is ignored and the portlet's default values are used for visitor scope attributes. If `visitorId` is specified (non-zero), `programId` must be specified; otherwise, `visitorId` is ignored.

Either `portletId` or `programId` must be specified (non-zero). If the value of `portletId` is 0, the value of `programId` must not be 0 because it is needed for retrieving `portletId`. If `programId` is specified, program-scope attributes are retrieved from the program; otherwise, the default values for the portlet are used.

BVI_EPPortletManager::setAttrsForVisitor()

Set one or more visitor specific dynamic attributes in a portlet for a visitor.

```
long setAttrsForVisitor(
    long portletId,
    long visitorId,
    long programId,
    BVI_MultiValueTable visitorMVTbl);
```

Parameters

`portletId`	Portlet ID
`visitorID`	Visitor ID
`programId`	Program ID
`visitorMVTbl`	Attributes to be set

Return value Returns 0 or an error code.

Portlet administration methods

This section describes BVI_EPPortletManager methods that portlet developers do not usually need to use. They are used in the Portlets area of the Admin Tool. If you plan to create your own administration tool, these methods may be useful to you.

BVI_EPPortletManager methods	Functionality
creator()	Creates a copy of the specified BVI_EPPortletManager object.
clone()	Returns a copy of the specified BVI_EPPortletManager object.
parseRegFile()	Parses a portlet XML registration file and put the data into a content object.
registerPortlet()	Loads the portlet data into the database.
deletePortlet()	Deletes the specified portlet.
reRegisterPortlet()	Reloads the portlet data into the database.
setAttrsForProgram()	Sets one or more program specific dynamic attributes in a portlet for a program.

BVI_EPPortletManager::creator()

Creates a copy of the specified BVI_EPPortletManager object.

JavaScript
```
BVI_EPPortletManager creator(BVI_EPPortletManager ref);
```

Java
```
public BVI_EPPortletManager creator (BVI_EPPortletManager ref);
```

Java exception Throws BVObjectNotCreatedException, BVWFactoryNotFoundError

Parameters

ref	An existing BVI_EPPortletManager object.

Return value A BVI_EPPortletManager object, or null if an error occurs.

```
creator( BVI_EPPortletManager ref );
```

BVI_EPPortletManager::clone()

Returns a copy of the specified BVI_EPPortletManager, or null on error.

JavaScript
```
BVI_EPPortletManager clone;
```

Java
```
public final BVI_EPPortletManager clone_J()
```

Java exception Throws BVWComponentException.

Return value A copy of this object, or null on error.

Example
```
var ptlt = new BVI_EPPortletManager;
var clone1 = ptlt.clone();
```

BVI_EPPortletManager::parseRegFile()

Parse a portlet XML registration file and put the data into a content object.

```
BVI_Content parseRegFile (string regFileName);
```

Parameters

regFileName	A string; the full path of the registration file name

Return value A BVI_Content including the following multi-value tables:

BV_EP_PTLT_ATTR	Portlet dynamic attributes.
BV_EP_PTLT_VALUE	Portlet dynamic attribute control values.

Remarks

The BVI_Content object contains all the generic attributes of the portlet (PTLT_NAME, PTLT_DESC, and so on. The multi-value table BV_EP_PTLT_ATTR contains metadata for all the portlet dynamic attributes (such as ATTR_SCOPE, CTRL_TYPE, and so on). The multi-value table BV_EP_PTLT_VALUE contains the list of control values specified in the registration file. Use these to populate select boxes and radio buttons.

BVI_EPPortletManager::registerPortlet()

Load the portlet data into the database.

```
long registerPortlet (
    BVI_Content portletCnt,
    BVI_MultiValueTable dynamicAttrTbl,
    BVI_MultiValueTable controlValueTbl);
```

Parameters

`portletCnt`	Portlet content (the OID is set after the data is loaded into the database).
`dynamicAttrTbl`	Portlet dynamic attribute table.
`controlValueTbl`	Portlet control value table.

Return value

Returns 0 or an error code.

Remarks

Both multivalue tables can have a null value. If the value of `dynamicAttrTbl` is null, the value of `controlValueTbl` must also be null.

BVI_EPPortletManager::deletePortlet()

Deletes the specified portlet. All related information in the portlet list tables, the visitor / portlet list table, and the program portlet list table are also removed.

```
long deletePortlet (long portletId);
```

Parameters

`portletId`	portlet ID

Return value

Returns 0 or an error code.

BVI_EPPortletManager::reRegisterPortlet()

Reloads the portlet data into the database.

```
long reRegisterPortlet (
    long portletId,
    BVI_Content portletCnt,
    BVI_MultiValueTable dynamicAttrTbl,
    BVI_MultiValueTable controlValueTbl);
```

Parameters

`portletId`	ID of the portlet to be re-registered.
`portletCnt`	Portlet content
`dynamicAttrTbl`	Portlet dynamic attribute table.
`controlValueTbl`	Portlet control value table.

Return value Returns 0 or an error code.

Remarks The related visitor-portlet and program-portlet list table are updated during re-registration.

BVI_EPPortletManager::setAttrsForProgram()

Sets one or more program specific dynamic attributes in a portlet for a program.

```
long setAttrsForProgram(
    long portletId,
    long programId,
    BVI_MultiValueTable programMVTbl);
```

Parameters

`portletId`	Portlet ID.
`programID`	Program ID.
`programMVTbl`	Attributes to be set.

Return value Returns 0 or an error code.

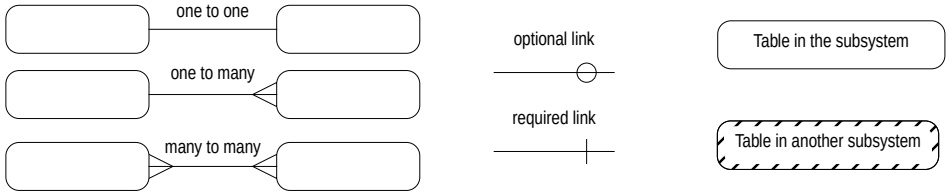
A Database schema

BroadVision InfoExchange Portal extends the One-To-One database schema with new attributes and tables in these categories:

- "InfoExchange Portal data types," next, lists the data types that are specific to InfoExchange Portal.
- "Channel and program tables" on page 284 describes the tables that contain the InfoExchange Portal channel and program information.
- "User profile and related tables" on page 291 describes the tables that contain the InfoExchange Portal user profile information.
- "Category content type tables" on page 306 describes tables that contain extensions to the Category content type in One-To-One Commerce.
- "Account profile content type tables" on page 309 describes tables that contain extensions to the Account profile content type in One-To-One Commerce.
- "Qualifier tables" on page 318 describes the tables that define qualifiers and qualifier values used in an InfoExchange Portal site.
- "Project tables" on page 320 describes the tables that contain information about projects, phases, tasks, and announcements.
- "Message attachment tables" on page 329 describes the table that holds message attachment file information.
- "Lead management tables" on page 330 describes the tables that hold sales lead information.
- "Page type tables" on page 335 describes the tables that hold page type, block, and program information for configurable home pages.
- "Portlet tables" on page 338 describes the tables that hold attribute and value information for portlets.

Your implementation of the database might require you to extend the database schema further. For detailed information about One-To-One database schema and for instructions on customizing it, see the *Database Schema Reference* and the *Database Administrator's Guide* manuals provided with the BroadVision One-To-One Enterprise.

The entity relationship diagrams in this chapter use the following notations:



InfoExchange Portal data types

InfoExchange Portal defines these data types in the file `ep_non_extendable_types.src`. When you install InfoExchange Portal these data types are added to the BroadVision One-To-One Enterprise BV_TYPE table.

Name	Semantics	Primitive	Mapping or range	Link	Max field length
OID_PROJECT	Object ID of the project.	OID		BV_EP_PROJECT	
OID_PHASE	Object ID of the phase.	OID		BV_EP_PHASE	
OID_GROUP	Object ID of the group.	OID		BV_EP_GROUP	
PTLT_TYPE_TYPE	Portlet Type.	ENUM	“JavaScript”, “JSP”		0
PTLT_SCOPE_TYPE	Portlet Scope Type.	ENUM	“site-wide”, “program”, “visitor”		0
PTLT_CTRL_TYPE	Portlet Control Type.	ENUM	“check box”, “radio button”, “select box”, “text”, “text area”		0
PTLT_OID_TYPE	Object ID of the project.	OID		BV_EP_PORTLET	
TASK_BASIC_TYPE	Task or Meeting.	ENUM	“Task” “Meeting”		0
DIST_NODE_TYPE	Distributed Node Type.	ENUM	“Channel” “User”		0
CHANNEL_PAGE_TYPE	Channel Page Type.	ENUM	“Script”, “URL”		0
PRG_PAGE_TYPE	Program Page Type.	ENUM	“Script”, “URL”		0
PRG_SRC_TYPE	Program Source Type.	ENUM	“Rule Set”, “Category”, “Content”, “Script”, “URL”, “Portlet”		0
OID_PROGRAM	Object ID of Program.	OID	Maps to a program in BV_KM_PROGRAM	BV_KM_PROGRAM	
KEY_EP_QUAL_VALUE	Object ID.	OID		EP_QUAL_VALUE	

InfoExchange Portal defines the following data types in the file `ep_extendable_types.src`.

Name	Semantics	Primitive	Mapping or range	Link	Max field length
METHOD	Contact method.	ENUM	“E-mail”, “Fax”, “Phone”		0
TASK_PRIORITY_TYPE	Task priority	ENUM	“High” “Medium” “Low”		0
TASK_STATUS_TYPE	Task status	ENUM	“Assigned” “Pending” “Completed”		0
ACCESS_TYPE	Project groups. (Obsolete)	ENUM	“Public” “Private”		0

You can change or add the enum choices for METHOD, TASK_PRIORITY_TYPE, and TASK_STATUS_TYPE and ACCESS_TYPE; you can also delete the enum choices for METHOD. See the *One-To-One Enterprise Database Administrator’s Guide* for details on adding new enum types.

In addition, InfoExchange Portal defines additional data types in the `mr_data_types.src` file. These are:

Name	Semantics	Primitive	Mapping or range	Link	Max field length
MR_INDUSTRY_TYPE	Industry	ENUM	"High Technology", "Telecommunications", "Utilities", "Government", "Manufacturing", "Advertising", "Creative Design", "Legal", "Retail", "Banking"		
MR_ACCOUNT_TYPE	Account Type	ENUM	"Buyer", "Supplier", "Buyer_and_Supplier", "Trading_Network", "Trading_Partner"		
MR_CO_RATING_TYPE	Company Rating	ENUM	"Excellent", "Good", "Fair", “Poor”		
MR_REGION_TYPE	Region	ENUM	“East”, “West”, “North”, “South”		
MR_LIFECYCLE_TYPE	Life Cycle State	ENUM	“Prospect”, “Non-qualified”, “Active”, “Suspended”		
HDR_TYPE_ENUM	Header	ENUM	“From”, “Sender”, “To”		
XML_TYPE_ENUM	XML type	ENUM	“cXML”, “CBL”, “OBI”, “OAG”, “BVXML”, “OCI”		

Name	Semantics	Primitive	Mapping or range	Link	Max field length
TRANS_TYPE_ENUM	Transaction type	ENUM	“CatalogRequestSetup”, “PurchaseOrder”, “RFQ”, “ShoppingCart”, “ReturnCart”, “PurchaseOrderAck”, “All”		
POST_PROT_ENUM	Document post format	ENUM	“http”, “https”		
MR_ATTACHM_USAGE	The type for usage	ENUM	“INTERNAL”, “EXTERNAL”		
OID_ORDERS	Object ID of Orders	OID		MR_ORDERS	
ORDER_TYPE_ENUM	Type of Order	ENUM	“Sales”, “Requisition”, “PurchaseOrder”, “RFQ”, “RFQResponse”, “RFQCounterResp”		0
ORDER_STATE_DICT	Order States	DICT	See “Mappings for the ORDER_STATE_DICT data type” on page 282.		
LOCATION_TYPE_DICT	Location Types	DICT	0 "United States" 1 "International"		
ITEM_STATE_DICT	Item state	DICT	0 "Valid" 1 "Cancel" 2 "Partial" 11"RFQGenerated" 12 "POGenerated" 20 "Closed"		
ITEM_CATALOG_ENUM	Item Catalog	ENUM	“InCatalog”, “OffCatalog”, “CatalogRequest”		0
MR_PACKAGE_TYPE	Package type	ENUM	"Receive", "Return"		
MR_PACKAGE_STATUS	Package status	ENUM	"In_progress", "Finished"		
MR_PKG_TRANS_TYPE	Package item transaction type	ENUM	"Receive", "Return", “Reject”		
MR_CONTRACT_OWNER_TYPE	Contract Owner Type	STRING	“Buyer”, “Host”		
MR_ORDER_PRIORITY_TYPE	Order Priority Type	STRING	“Regular”, “Urgent”		
PWA_CONTRACT_TYPE	Contract Type	ENUM	“Standard”, “Blanket”		
PWA_ITEM_DISC_TYPE	Item Discount Type	ENUM	“Unit”, “Volume”		
SLIST_TYPE_ENUM	Type of shopping list	ENUM	“User”, “Account”, “Store”		50
MR_INTEREST_TYPE	Product Interest Type	STRING	“Home Office”, “Office Furniture”, “Computers”, “Printers”		0

Name	Semantics	Primitive	Mapping or range	Link	Max field length
MR_PROD_DEPT_TYPE	Product Department Type	STRING	“Computers”, “Desk Accessories”, “Furniture”, “Office Equipment”, “Paper Supplies”, “Printer Accessories”, “Printers”, “Writing Instruments”		0
MR_WF_APPR_TASK_STATUS	Approval Task Status	DICT	0 “ApprovalPending” 1 “Approved” 2 “Forwarded” 3 “Notified” 4 “NotificationACK” 50 “Denied” 60 “ForcedApprove” 70 “CancelPending” 80 “AdminActionRequired”		
MR_WF_APPR_PROC_STATE	Approval Process State	ENUM	“UserAction”, “DaemonAction”, “Completed”		0
MR_WF_FUNCTION_TYPE	Function type	ENUM	“Criterion”, “Action”		
OID_PRICING_PGM	Object ID of Pricing Program	OID		MR_PRICING_PGM	
MR_RELATION_TYPE	Relationship Type	STRING	“CrossSell”, “UpSell”, “TechSpec”, “Substitute”		0
KEY_PRODUCT	key of related product	OID		PRODUCT	
KEY_MR_FEATURE_TYPE	key of feature type content	OID		MR_FEATURE_TYPE	
OID_FEATURE_TYPE	Object ID of feature types	OID		MR_FEATURE_TYPE	
MR_PROD_GRP_TYPES	Product Group Types	ENUM	“Product”, “Bundle”, “Product Group”, “Bundle Group”		
OID_SELECTION_PGM	Object ID of Product Selection Program	OID		MR_SELECTION_PGM	
HOLD_STATE_ENUM	Hold state for an item	ENUM	“Hold”, “Release”		0
HOLD_REASON_ENUM	Hold reason for an item	ENUM	“Contract”, “Account”, “Product”, “Out-of-catalog”		0

Mappings for the ORDER_STATE_DICT data type

The data type ORDER_STATE_DICT in the `mr_data_types.src` file contains the following mappings:

0	"UnknownOrderState"	"Unknown Order State"
1	"OrderNew"	"New Order"
2	"Being Authorized"	"Payment is being authorized"
3	"WaitFulfillment"	"Order is waiting to be fulfilled"
4	"BeingFulfilled"	"Order is being fulfilled"
5	"OrderFulfilled"	"Order is completely fulfilled"
6	"OrderPartiallyFulfilled"	"Order is partially fulfilled"
7	"WaitReturn"	"Payment is waiting to be return / credit back"
8	"BeingFullySettled"	"Payment is being fully settled"
9	"BeingPartiallySettled"	"Payment is being partially settled"
10	"BeingReturnSettled"	"Payment is being return settled"
11	"OrderComplete"	"Order is complete (fully settled)"
12	"OrderPartiallyComplete"	"Order is partially complete / settled"
13	"OrderReturnComplete"	"Payment return / credit complete successfully"
14	"OrderCancelled"	"Order is cancelled"
15	"BeingAuthCaptured"	"Order is being Auth-Captured"
16	"OrderAuthCaptured"	"Order is Auth-Captured"
17	"OrderPendingReturn"	"Order is being returned"
50	"DraftRequisition"	"Draft Requisition"
51	"NewRequisition"	"New Requisition"
52	"ReqBeingApproved"	"Requisition is being Approved"
53	"ReqApproved"	"Requisition is Approved"
54	"POBeingGenerated"	"Purchase Order (PO) is being generated"
55	"POGenerated"	"Purchase orders generated"
56	"POReceiving"	"Purchase orders on receive"
60	"RFQDraft"	"Draft RFQ"
61	"RFQSubmitted"	"RFQ Submitted"
62	"RFQResponded"	"RFQ Responded"
63	"RFQBidClosed"	"RFQ Bid Closed"
64	"RFQSelected"	"RFQ Selected"
65	"RFQCompleted"	"RFQ Completed"
66	"RFQPOCreated"	"PO Generated"
67	"RFQCancelled"	"RFQ Canceled"
70	"NewPurchaseOrder"	"New Purchase Order"
71	"POSent"	"PO Sent to exchange"

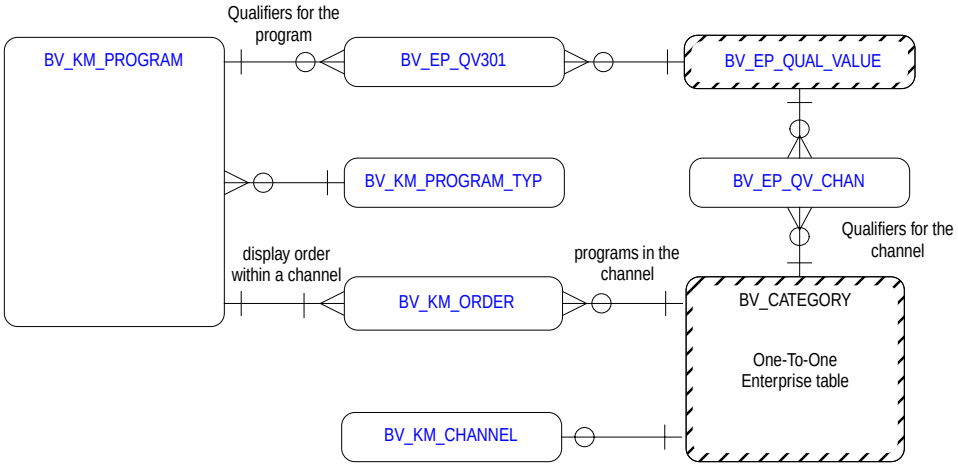
72	"ResponseAdded"	"Response received and added"
73	"ResponseChanged"	"Response received and changed"
74	"ResponseNoAction"	"Response received and no action"
75	"ResponseAcceptedAsIs"	"Response received and accepted as is"
76	"ResponseNotAccepted"	"Response received and not accepted"
77	"SupplierRejectACK"	"Supplier rejection acknowledged"
78	"PORevised"	"Purchase order revised"
85	"TaxAuditLogging"	"Being tax audit logged"
86	"TaxAuditLoggingComplete"	"tax audit logging complete"
87	"ErrorTaxLogging"	"Error during audit logging"
101	"AuthorizationError"	"Cannot authorize payment"
102	"FulfillmentError"	"Cannot fufill order"
103	"SettlementError"	"Cannot settle order"
104	"OrderReturnError"	"Payment return / credit back failed"
105	"OrderAuthCaptureError"	"Order Auth-Capture failed"
151	"ReqRejected"	"Requisition is Rejected"
152	"POGenerationFailed"	"Purchase Order (PO) Generation failed"
153	"NewReqWithNoWorkFlow"	"New Requisition BUT with NO workflow"
154	"ReqCanceled"	"Requisition is canceled by request"
171	"POSentFailure"	"Failed to send PO to exchange"
172	"POTransformFailure"	"Failed to generate XML for PO"
173	"POServerRejection"	"PO document rejected from server before reaching supplier"
174	"POItemsUpdateFailure"	"Failed to update the priced items when processing a PurchaseOrderResponse"
175	"POCancel"	"PO Cancelled by request"
176	"POClosed"	"Purchase orders closed"
301	"SupplierReceived"	"Supplier received the order"
302	"RequestForCancel"	"Request to cancel order"
305	"Shipped"	"Order shipped"
306	"Invoiced"	"Order invoiced"
307	"Paid"	"Order paid"
501	"XMLPONew"	"Order received in XML from an external entity"
502	"XMLPOTaxShipCalc"	"Tax and Shipping Costs calculated"
503	"XMLPOTaxShipCalcError"	"Error while computing tax and shipping costs"

999	"UnknownError"	"Unknown Error"
1001	"FirstUserDefinableState"	"First User Definable state"

Channel and program tables

These tables contain the InfoExchange Portal channel and program information.

- BV_KM_CHANNEL contains the channel definitions; this table is a child of the One-To-One Enterprise BV_CATEGORY table. Channels are implemented as categories in the content type KM_PROGRAM.
- BV_EP_QV_CHAN lists the qualifiers for a channel. This table is a related attributes list, and a child of the One-To-One Enterprise BV_CATEGORY table. The possible navigation filters for the site are defined in BV_EP_QUAL_VALUE.
- BV_KM_ORDER lists the programs in a channel, and defines the order that those programs appear when displayed together in a list. This table is a child of the One-To-One Enterprise BV_CATEGORY table.
- BV_KM_PROGRAM holds the program definitions for the site.
- BV_KM_PROGRAM_TYPE defines the program types that may be used in the site.
- BV_EP_QV301 lists the qualifiers for a program. This table is a related attributes list, and a child of BV_KM_PROGRAM. The possible qualifiers for the site are defined in BV_EP_QUAL_VALUE.



BV_KM_CHANNEL

Contains the InfoExchange Portal channel definitions. This table is a related attributes list, and a child of the One-To-One Enterprise BV_CATEGORY table.

Schema file	`ep_category_content_ext.src`

Table	BV_KM_CHANNEL	This name is required; do not change it.
Parent	BV_CATEGORY	The category in the KM_PROGRAM content type representing a channel.
Group	Channel Information	In the One-To-One Enterprise category schema.

All of the following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
CHANNEL_NAME	STRING	varchar(80)	KEY, REQ_ATTR	Channel Name.
CHANNEL_PAGE_TYPE	CHANNEL_PAGE_TYPE	int	REQ_ATTR	URL or Script.
CHANNEL_PAGE_PATH	STRING	varchar(255)	—	Path for URL or Script.
CHANNEL_DESC	TEXT	—	—	Channel Description.
CHANNEL_MGR_NAME	STRING	varchar(80)	—	Channel Manager Name.
CHANNEL_MGR_EMAIL	STRING	varchar(80)	—	Channel Manager's E-mail address.
CHANNEL_ICON_PATH	FILE_PATH	varchar(128)	—	Channel Icon Path. Not used in InfoExchange Portal 6.0.0.
TOGGLE_ICON_PATH	FILE_PATH	varchar(128)	—	Channel Toggle Icon Path. Not used in InfoExchange Portal 6.0.0.

BV_EP_QV_CHAN

Lists the qualifiers for a channel. This table is a related attributes list, and a child of the One-To-One Enterprise BV_CATEGORY table. The possible navigation filters for the site are defined in BV_EP_QUAL_VALUE.

Schema file	`ep_category_content_ext.src`	
Table	BV_EP_QV_CHAN	This name is required; do not change it.
Parent	BV_CATEGORY	The category representing the channel that has these qualifiers.
Related to	BV_EP_QUAL_VALUE	Defines the qualifiers available in the site.
Group	Qualifier Values on Channel	In the One-To-One Enterprise category schema.

All of the following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
QID	LONG, NOT_NULL		KEY, HIDDEN, REQ_ATTR	Qualifier ID
QVID	LONG, NOT_NULL		KEY, REQ_ATTR	Qualifier Value ID

BV_KM_ORDER

Lists the programs in a channel, and defines the order in which those programs appear when displayed together in a list. This table is a related attributes list, and a child of the One-To-One Enterprise BV_CATEGORY table.

Schema file	`ep_category_content_ext.src`	
Table	BV_KM_ORDER	This name is required; do not change it.
Parent	BV_CATEGORY	The category that represents the channel that uses this set of programs.
Related to	BV_KM_PROGRAM	The programs.
Group	Program Order	In the One-To-One Enterprise category schema.

All of the following attributes are required.

Name	Type	Attribute Kind	Friendly name or Semantics
PROGRAM_OID	OID_PROGRAM	KEY, REQ_ATTR	Program
DISPLAY_ORDER	LONG	—	Display Order of Program within Channel

BV_KM_PROGRAM

Contains the definitions of the site's InfoExchange Portal programs. There are currently two BV_KM_PROGRAM files. The first schema file is located as follows:

Schema file	`ep_program_base.src`	
Table	BV_KM_PROGRAM	This name is required; do not change it

Class	KM_PROGRAM	This table is hidden; it may not be viewed directly from the Command Center
Service	ALL_STORE	This table is available to the entire site
Content	KM_PROGRAM “One-to-One Knowledge Program”	Type 301

All of the following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
“Basics” group				
PROGRAM_NAME	STRING NOT_NULL	varchar(80)	REQ_ATTR	Program Name
STORE_ID	LONG NOT_NULL	—	REQ_ATTR, RONLY, HIDDEN	Interactive Service ID (Read-only)
CREATION_TIME	DATETIME NOT_NULL	—	REQ_ATTR, RONLY	Date/Time Program was Created (Read-only)
STATUS	STATUS_TYPE NOT_NULL	—	REQ_ATTR	Available/Unavailable for Display on Web Site
DELETED	LONG NOT_NULL	—	REQ_ATTR, RONLY, HIDDEN	Remove From/Keep in Database
LAST_MOD_TIME	DATETIME NOT_NULL	—	REQ_ATTR, RONLY	Date/Time of Last Modification to Program (Read-only)
“Details” group				
PROGRAM_DESC	TEXT	—	—	Program Description
PROGRAM_MGR_NAME	STRING	varchar(80)	—	Program Manager Name
PROGRAM_MGR_EMAIL	STRING	varchar(80)	—	Program Manager E-mail
PROGRAM_TYPE	LONG NOT_NULL	—	REQ_ATTR	Program Type (the OID in BV_KM_PROGRAM_TYPE)
PROGRAM_PAGE_TYPE	PRG_PAGE_TYPE NOT_NULL	int	REQ_ATTR	Program Page Type
PROGRAM_PAGE_PATH	STRING	varchar(255)	—	URL or Relative Path to Script File
PROGRAM_ICON_PATH	FILE_PATH	varchar(128)	—	Relative Path to Icon Image File Not used in InfoExchange Portal 6.0.0.
TOGGLE_ICON_PATH	FILE_PATH	varchar(128)	—	Relative Path to Highlighted Icon Image File Not used in InfoExchange Portal 6.0.0.

The second schema file extends BV_KM_PROGRAM and is located as follows:

Schema file	`ep_program_ext.src`	
Table	BV_KM_PROGRAM	This name is required; do not change it.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
"Source of Content for Program" group				
PROGRAM_SRC_TYPE	PRG_SRC_TYPE	int	-	Type of Content Source, such as Rule Set, Category or Content
PROGRAM_SRC_ID	LONG	-	-	ID of the Content Source
SUBCAT_INCLUDED	BOOLEAN_ENUM	-	-	If Content Source is a Category, Include Content in Subcategories
SHOW_ONLY_READABLE	BOOLEAN_ENUM	-	-	Show Only Readable Content
OTHER1	STRING	varchar(100)	-	Additional Program Information
OTHER2	STRING	varchar(100)	-	Additional Program Information
"Embedded Content for Program" group				
EMBED_SCRIPT_PATH	STRING	varchar (255)	-	URL or Relative Path to Script File.
INTCOL1	LONG	-	-	Additional integer 1, for customer use.
INTCOL2	LONG	-	-	Additional integer 2, for customer use.
INTCOL3	LONG	-	-	Additional integer 3, for customer use.
STRCOL1	STRING	varchar (255)	-	Additional text 1, for customer use.
STRCOL2	STRING	varchar (255)	-	Additional text 2, for customer use.
STRCOL3	STRING	varchar (255)	-	Additional text 3, for customer use.
"Publishing Category and Script for Program" group				
PUB_CATEGORY_ID	LONG	-	KEY, REQ_ATTR	ID of the publish category.
PUB_SCRIPT_PATH	STRING	varchar (255)	-	URL or Relative Path to Script File.
"Primary Parent for Program" group				
PRIMARY_PARENT_ID	LONG	-	-	Program Primary Parent ID.
"Program Text Cache Settings" group				
CACHEABLE	LONG	-	-	Program cacheable.
TIMEOUT	LONG	-	-	Program Text Cache timeout value.
"Portlet Attribute for Program" group				
PTLT_ID	PTLT_OID_TYPE NOT_NULL		KEY, REQ_ATTR	Portlet Object ID.
ATTR_ID	STRING	varchar(10)	KEY, REQ_ATTR	Portlet Attribute ID.
ATTR_VALUE	STRING	varchar (128)	-	Portlet Attribute Value.

BV_KM_PROGRAM_TYPE

Defines all the program types that may be used in the site.

Schema file	ep_program_type_base.src	
Table	BV_KM_PROGRAM_TYPE	This name is required; do not change it
Related to	BV_KM_PROGRAM	The programs that are of these types.
Class	KM_PROGRAM_TYPE	This table is hidden; it may not be viewed directly from the Command Center
Service	ALL_STORE	This table is available to the entire site
Content	KM_PROGRAM_TYPE “One-to-One Knowledge Program Type”	Type 302

All of the following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
“Basics” group				
TYPE_NAME	STRING NOT_NULL	varchar(80)	REQ_ATTR	Program Type Name
STORE_ID	LONG NOT_NULL	—	REQ_ATTR, RONLY, HIDDEN	Interactive Service ID (Read-only)
CREATION_TIME	DATETIME NOT_NULL	—	REQ_ATTR, RONLY	Date/Time Program Type was Created (Read-only)
STATUS	STATUS_TYPE NOT_NULL	—	REQ_ATTR	Available/Unavailable for Use on Web Site
DELETED	LONG NOT_NULL	—	REQ_ATTR, RONLY, HIDDEN	Remove From/Keep in Database
LAST_MOD_TIME	DATETIME NOT_NULL	—	REQ_ATTR, RONLY	Date/Time of Last Modification to Program Type (Read-only)
“Details” group				
PROGRAM_TYPE_DESC	STRING	varchar(255)	—	Program Type Description
PROGRAM_CNT_TYPE	LONG NOT_NULL	—	REQ_ATTR	Content Type of Programs’ Content Source
RULE_SET_CATEGORY	LONG	—	—	Category of Rule Sets Available to Programs
TYPE_ICON	FILE_PATH	varchar(128)	—	Relative Path to Icon Image File Not used in InfoExchange Portal 6.0.0.
TOGGLE_ICON	FILE_PATH	varchar(128)	—	Relative Path to Highlighted Icon Image File Not used in InfoExchange Portal 6.0.0.

BV_EP_QV301

Lists the qualifiers for a program. This table is a related attributes list, and a child of BV_KM_PROGRAM. The possible qualifiers for the site are defined in BV_EP_UPROF_QVAL.

Table	BV_EP_QV301	This name is required; do not change it.
Related to	BV_KM_PROGRAM	The programs that have these qualifiers.
Related to	BV_EP_QUAL_VALUE	Defines the qualifiers available in the site.

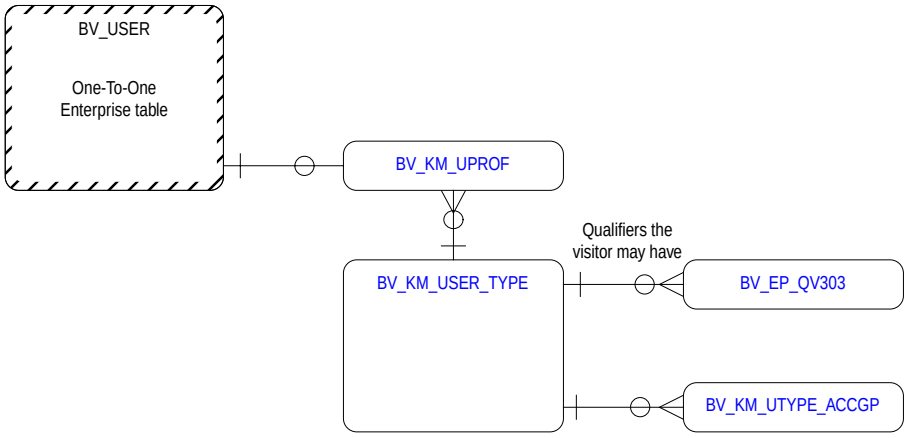
All of the following attributes are required.

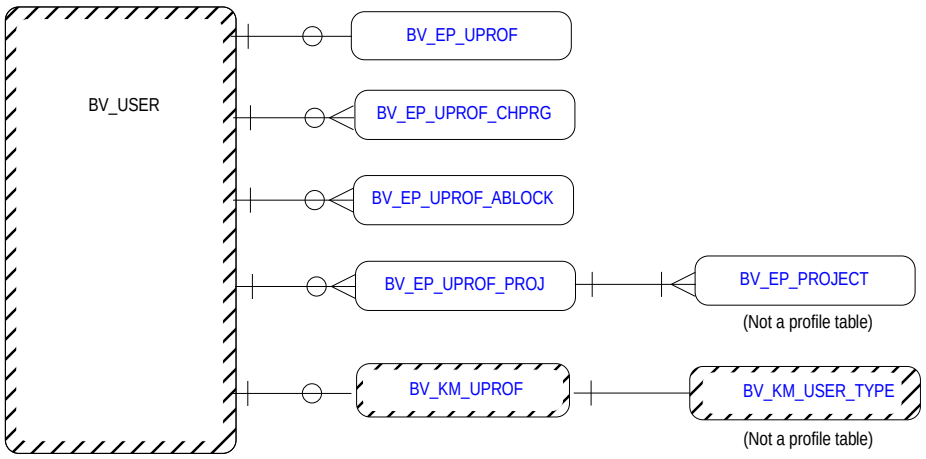
Name	Type	Column	Attribute Kind	Friendly name or Semantics
QID	LONG, NOT_NULL		KEY, HIDDEN, REQ_ATTR	Qualifier ID
QVID	LONG, NOT_NULL		KEY, REQ_ATTR	Qualifier Value ID

User profile and related tables

These tables contain the InfoExchange Portal user profile information.

- BV_KM_UPROF defines visitor profile attributes specific to InfoExchange Portal. This table is a related attributes list, and a child of the One-To-One Enterprise BV_USER table.
- BV_EP_UPROF_QVAL lists the qualifiers assigned to an InfoExchange Portal user account. This table is a related attributes list, and a child of the One-To-One Enterprise BV_USER table. The possible qualifiers for the site are defined in BV_EP_QUAL_VALUE.
- BV_EP_UPROF defines visitor profile attributes specific to InfoExchange Portal. This table is a related attributes list, and a child of the One-To-One Enterprise BV_USER table.
- BV_EP_UPROF_BLOCK holds the configurable blocks in the home page that appear to the visitor.
- BV_EP_UPROF_CHPRG identifies which programs within the blocks will appear to the user.
- BV_EP_UPROF_ABLOCK holds the expand and collapse state of the required blocks for a user.
- BV_EP_UPROF_PROJ lists the projects for users.
- BV_KM_USER_TYPE defines the InfoExchange Portal user templates. Each user account in the BV_KM_UPROF has an associated user template defined in this table.
- BV_KM_UTYPE_ACCGP lists the default access groups of a user template. This table is a related attributes list, and a child of BV_KM_USER_TYPE.
- BV_EP_QV303 lists the qualifiers for a user template. This table is a related attributes list, and a child of BV_KM_USER_TYPE. The possible qualifiers for the site are defined in BV_EP_QUAL_VALUE.
- BV_EP_UPROF_SRVAD lists the services administered by a specific visitor.
- BV_EP_UPROF_DISTAD lists the information needed for distributed channel and user administration.
- BV_EP_UPROF_PTLT lists the portlet attributes for a visitor.
- BV_KM_SUBSCRIBED contains personal subscriptions (bookmarks) of individual visitors.
- BV_KM_SUB_DEFAULTS contains default subscriptions (bookmarks) for user templates.





The following tables contain user profile attributes used for organization entities. These attributes also belong to the user profile schema:

- “BV_USER_PROFILE” on page 302
- “BV_MR_USER_PROFILE” on page 302
- “MR_ADD_BOOK” on page 303
- “MR_INTERESTS” on page 303
- “MR_PURCHASED_ITEMS” on page 304
- “MR_ORDER_HISTORY” on page 304
- “MR_PERSISTENT_CART” on page 304
- “MR_DELEGATEES” on page 305
- “BV_MR_USER_ACCT” on page 305
- “BV_MR_ACCT_ROLE” on page 305
- “MR_PC_ITEM_PROPS” on page 306
- “MR_USER_ORG” on page 306
- “MR_TAX_UPROF” on page 306

BV_KM_UPROF

Defines visitor profile attributes specific to InfoExchange Portal. This table is a related attributes list, and a child of the One-To-One Enterprise BV_USER table.

Schema file	**`ep_prof_ext.src`**	
Table	BV_KM_UPROF	This name is required; do not change it.
Parent	BV_USER	The user account to which this profile applies.
Related to	BV_KM_USER_TYPE	InfoExchange Portal user template.
Group	Knowledge Attributes	In the One-To-One Enterprise visitor schema.

All of the following required attributes are required. You may add additional attributes, but you may not change the attributes listed in the following table.

Name	Type	Semantics or Friendly name
KM_USERTYPE_ID	LONG	User Template ID
KM_USER_ADMIN	BOOLEAN_ENUM	Is User Admin?
KM_CHANNEL_ADMIN	BOOLEAN_ENUM	Is Channel Admin?
KM_USER_DELETED	BOOLEAN_ENUM	If True, the visitor is not allowed to log into the InfoExchange Portal site.
KM_SITE_ADMIN	BOOLEAN_ENUM	Is Site Admin
KM_EMAIL_ALERT	BOOLEAN_ENUM	Wants e-mail alerts for new content

BV_EP_UPROF_QVAL

Contains the qualifiers assigned to an InfoExchange Portal visitor. This table is a related attributes list and a child of the One-To-One Enterprise BV_USER table. The possible qualifiers for the site are defined in BV_EP_QUAL_VALUE.

Schema file	`ep_uprof_qual_val_ext.src`	
Table	BV_EP_UPROF_QVAL	This name is required; do not change it.
Parent	BV_USER	The user account to which this profile applies.
Related to	BV_EP_QUAL_VALUE	Defines the qualifiers available in the site.

All of the following attributes are required. You may add additional attributes, but you may not change the attributes listed in the following table.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
QID	LONG, NOT NULL		KEY, HIDDEN, REQ_ATTR	Qualifier ID.
QVID	LONG, NOT NULL		KEY, HIDDEN, REQ_ATTR	Qualifier Value ID.

BV_EP_UPROF

Defines visitor profile attributes specific to InfoExchange Portal. This table is a related attributes list, and a child of the One-To-One Enterprise BV_USER table.

Schema file	`ep_prof_ext.src`	
Table	BV_EP_UPROF	This name is required; do not change it.
Extension of	BV_USER	User to which this profile applies.
Group	ERM App Attributes	In the InfoExchange Portal visitor schema..

Child	BV_EP_UPROF_CHPRG	Programs within blocks pertaining to the user.
Child	BV_EP_UPROF_ABLOCK	Required blocks pertaining to the user.
Child	BV_EP_UPROF_PROJ	Projects for users.

All of the following attributes except the example lead management matching attributes are required. You may add additional attributes, but you may remove only the example closed-loop process management attributes.

Name	Type	Column	Semantics or Friendly name
Configurable home page attributes (required)			
EP_HEADING	STRING	varchar (20)	Visitor's choice of block header color.
EP_SUBHEADING	STRING	varchar (20)	Visitor's choice of topic header color (topics in a block are programs).
Project attributes (required)			
EP_PROJ_GROUPS	STRING	varchar (255)	Project groups to which the user belongs (OBSOLETE).
EP_PROJECT_ADMIN	BOOLEAN_ENUM	-	Is user a project admin? That is, is the user allowed to create projects?
EP_DEF_PROJECT	LONG	-	Project OID for visitor's choice of current project (BV_EP_PROJECT)
Closed-Loop process management attributes (required)			
EP_OWNER_ADMIN	BOOLEAN_ENUM	-	Is user a lead owner admin? That is, is the user allowed to manage lead owner organizations (such as resellers)?
Example closed-Loop process management matching attributes (optional)			
EP_PRODUCT	STRING	varchar (80)	Product ID of product referenced by a lead.
EP_QUANTITY	LONG	-	Quantity of product requested in a lead.

BV_EP_UPROF_BLOCK

Holds the configurable blocks for a visitor. This table is a list of the blocks in the home page that appear to the visitor.

Schema file	`ep_uprof_conf_blocks_ext.src`	
Table	BV_EP_UPROF_BLOCK	This name is required; do not change it.
Parent	BV_EP_UPROF	User to which these blocks apply.
Related to	BV_EP_BLOCK	Block content.
Group	ERM App Configurable Block Attributes	In the InfoExchange Portal visitor schema..

All of the following attributes are required.

Name	Type	Attribute Kind	Friendly name or Semantics
EP_BLOCK_ID	LONG	KEY, REQ_ATTR	The OID of the block the visitor chose to be visible in the home page. (BV_EP_BLOCK)
EP_PAGETYPE_ID	LONG	REQ_ATTR	The OID of the page type containing the block (BV_EP_PAGE_TYPE).
EP_COLUMN_ID	LONG	REQ_ATTR	Column number that the block is displayed in on the home page (first column is 1).
EP_DISPLAY_ORDER	LONG	REQ_ATTR	Display order of the block within a column (the first block has a display order of 0).
EP_EXPAND	LONG	REQ_ATTR	Expand flag: 0 means the block is collapsed in the home page; 1 means the block is expanded in the home page.

BV_EP_UPROF_CHPRG

This table holds information identifying which programs within the blocks are to be displayed to the visitor.

Schema file	ep_uprof_conf_chprg_ext.src	
Table	BV_EP_UPROF_CHPRG	This name is required; do not change it.
Parent	BV_EP_UPROF	User to which these blocks and programs apply.
Related to	BV_EP_BLOCK	Block content.
Group	ERM App Configurable Channel/Program Attributes	In the InfoExchange Portal visitor schema..

All of the following attributes are required.

Name	Type	Attribute Kind	Friendly name or Semantics
EP_CHPRG_ID	LONG	KEY, REQ_ATTR	The OID of the program to be displayed to the user in a block of the home page.
EP_BLOCK_ID	LONG	KEY, REQ_ATTR	The OID of the block containing the program (BV_EP_BLOCK).
EP_PAGETYPE_ID	LONG	REQ_ATTR	The OID of the page type containing the block (BV_EP_PAGE_TYPE).
EP_ENTRY_NUM	LONG	REQ_ATTR	Maximum number of entries to display in this program on the home page.
EP_DISPLAY_ORDER	LONG	REQ_ATTR	Display order of this program within its block.

BV_EP_UPROF_ABLOCK

This table holds the expand and collapse state of the required blocks for a visitor.

Schema file	ep_uprof_conf_ablocks_ext.src	
Table	BV_EP_UPROF_ABLOCK	This name is required; do not change it.
Parent	BV_EP_UPROF	User to which these blocks apply.
Related to	BV_EP_BLOCK	Block content.
Group	ERM App Attributes of Required Blocks	In the InfoExchange Portal visitor schema..

The following attributes are required. This table is a multi-value attribute table for holding the Required (admin) blocks for a visitor.

Name	Type	Attribute Kind	Friendly name or Semantics
EP_BLOCK_ID	LONG	KEY, REQ_ATTR	Block ID (BV_EP_BLOCK)
EP_EXPAND	LONG	REQ_ATTR	Expand flag 0 means the block is collapsed 1 means the block is expanded

BV_EP_UPROF_PROJ

Lists the projects for users.

Schema file	ep_participant_ext.src	
Table	BV_EP_UPROF_PROJ	This name is required; do not change it.
Parent	BV_EP_UPROF	Profile table.
Related to	BV_EP_PROJECT	Project table.
Group	Projects for Users	In the InfoExchange Portal visitor schema..

Name	Type	Column	Attribute Kind	Friendly name or Semantics
EP_PROJECT_OID	OID_PROJECT	-	KEY, REQ_ATTR	The OID of the project in which the user participates (BV_EP_PROJECT).
EP_PROJECT_CONT	BOOLEAN_ENUM	-	-	Is user a contact for the project?
EP_PROJECT_OWNER	BOOLEAN_ENUM	-	-	Is user a project owner?
EP_CONTACT_INFO	STRING	varchar (80)	-	Contact information, such as the role the user has in the project.

BV_KM_USER_TYPE

Defines the InfoExchange Portal user templates. Each user account in the BV_KM_UPROF has an associated user template defined in this table.

Schema file	`ep_user_type_base.src`, `ep_user_type_ext.src`	
Table	BV_KM_USER_TYPE	This name is required; do not change it.
Related to	BV_KM_UPROF	The visitor's profile attributes.
Child	BV_KM_UTYPE_ACCGP	Lists the default access groups for this user template.
Child	BV_EP_QV303	Lists the qualifiers for this user template.
Class	KM_USER_TYPE	This table is hidden; it may not be viewed directly from the Command Center
Service	ALL_STORE	This table is available to the entire site
Content	KM_USER_TYPE "One-to-One Knowledge User Template"	Type 303

Defines the InfoExchange Portal extensions to the BV_KM_USER_TYPE table. This is not actually a user profile table, but rather an additional content type. It is shown here in reference to its usage through the BV_USER table.

Schema file	`ep_user_type_ext.src`	
Table	BV_KM_USER_TYPE	InfoExchange Portal user template.
Related to	BV_USER	User information.

All of the following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
"Basics" group (`ep_user_type_base.src`)				
TYPE_NAME	STRING NOT_NULL	varchar(80)	REQ_ATTR	User Type Name
STORE_ID	LONG NOT_NULL	—	REQ_ATTR, RONLY, HIDDEN	Interactive Service ID (Read-only)
CREATION_TIME	DATETIME NOT_NULL	—	REQ_ATTR, RONLY	Date/Time of Last Modification to User Type (Read-only)
STATUS	STATUS_TYPE NOT_NULL	—	REQ_ATTR	Available/Unavailable for Use on Web Site
DELETED	LONG NOT_NULL	—	REQ_ATTR, RONLY, HIDDEN	Remove From/Keep in Database
LAST_MOD_TIME	DATETIME NOT_NULL	—	REQ_ATTR, RONLY	Date/Time of Last Modification to User Type (Read-only)

Name	Type	Column	Attribute Kind	Friendly name or Semantics
"Details" group (ep_user_type_ext.src)				
USER_TYPE_DESC	STRING	varchar(255)	—	User Type Description
HOME_PAGE	FILE_PATH	varchar(128)	—	Relative Path to Home Page Script
PERSONAL_PROF_PAGE	FILE_PATH	varchar(128)	—	Relative Path to Personal Profile Script
MATCHING_PROF_PAGE	FILE_PATH	varchar(128)	—	Relative Path to Matching Profile Script
PAGETYPE_ID	LONG		REQ_ATTR	Home Page Type ID for the page type used for all users in this user template (BV_EP_PAGE_TYPE).
EP_PROJ_GROUPS	STRING	varchar(255)		Project groups to which the user belongs (OBSOLETE).

BV_KM_UTYPE_ACCGP

Lists the default Publishing Center access groups to which an InfoExchange Portal visitor in this user template belongs. This table is a related attributes list, and a child of BV_KM_USER_TYPE.

Schema file	ep_user_type_ext.src	
Table	BV_KM_UTYPE_ACCGP	This name is required; do not change it.
Parent	BV_KM_USER_TYPE	User template to which this access group applies.
Group	User Type Access Groups	In the InfoExchange Portal user type schema.

The following attribute is required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
ACCESS_GROUP	LONG	KEY, REQ_ATTR	—	Access Group

BV_EP_QV303

Lists the qualifiers for a user template. This table is a related attributes list, and a child of BV_KM_USER_TYPE. The possible qualifiers for the site are defined in BV_EP_QUAL_VALUE.

Table	BV_EP_QV303	This name is required; do not change it.
Parent	BV_KM_USER_TYPE	User template for which these qualifiers apply.
Related to	BV_EP_QUAL_VALUE	Defines the qualifiers available in the site.

All of the following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
QID	LONG, NOT_NULL		KEY, HIDDEN, REQ_ATTR	Qualifier ID
QVID	LONG, NOT_NULL		KEY, REQ_ATTR	Qualifier Value ID

BV_EP_UPROF_SRVAD

Lists the services administered by a specific visitor.

Schema file	`ep_uprof_srv_ext.src`	
Table	BV_EP_UPROF_SRVAD	This name is required; do not change it.
Parent	BV_KM_USER_TYPE	User template to which this list applies.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
EP_ADMIN_SRV_ID	LONG	-	KEY, REQ_ATTR	Service ID; an InfoExchange Portal service the visitor administers.

BV_EP_UPROF_DISTAD

Lists the information needed for distributed channel and user administration.

Schema file	`ep_uprof_distad_ext.src`	
Table	BV_EP_UPROF_DISTAD	This name is required; do not change it.
Parent	BV_KM_USER_TYPE	User template to which this information applies.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
EP_DIST_NOD_OID	LONG NOT NULL	-	KEY, HIDDEN, REQ_ATTR	A node the visitor administer.
EP_DIST_SRV_ID	LONG	-	HIDDEN	Service ID.
EP_DIST_NOD_TYPE	BOOLEAN_ENUM int		HIDDEN, REQ_ATTR	IChannel (0) or User (1) Node.

BV_EP_UPROF_PTLT

Lists the portlet attributes for a visitor.

Schema file	`ep_uprof_ptlt_ext.src`	
Table	BV_EP_UPROF_PTLT	This name is required; do not change it.
Parent	BV_USER_PROFILE	Portlet attributes and metadata.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
PTLT_ID	PTLT_OID_TYPE, NOT_NULL		KEY, REQ_ATTR	Portlet Object ID.
ATTR_ID	STRING	varchar(10)	KEY, REQ_ATTR	Portlet Attribute ID. The PTLT_ID and ATTR_ID are a composite key. The ATTR_ID must be unique within a portlet.
PROG_ID	PTLT_OID_TYPE, NOT_NULL		KEY, REQ_ATTR	Program ID.
ATTR_VALUE	STRING	varchar(128)		Portlet Attribute Value.

BV_KM_SUBSCRIBED

Contains bookmarks of individual visitors.

Table	BV_KM_SUBSCRIBED	This name is required; do not change it.
Parent	BV_USER_PROFILE	Portlet attributes and metadata.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
USER_ID	INT, NOT_NULL			User ID.
SUB_NAME	STRING	varchar2(80)		Friendly name of this subscription (bookmark)..
SUB_DESC	STRING	varchar2(255)		Description of this subscription (bookmark).
OBJECT_TYPE	INT, NOT_NULL			Subscribed object type : 0: unclassified 1: channell 2: program 3: category 4: content
SERVICE_ID	INT			Service ID.
CONTENT_TYPE	INT			Content Type.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
SYSTEM_ID	INT			System ID.
CHANNEL_ID	INT			Channel ID.
PROGRAM_ID	INT			Program ID.
CATEGORY_ID	INT			Category ID.
CONTENT_ID	INT			Content ID.
DISPLAY_ORDER	INT			Relative order among subscription (bookmark) list.

BV_KM_SUB_DEFAULTS

Contains default bookmarks for user templates.

Table	BV_KM_SUBSCRIBED	This name is required; do not change it.
Parent	BV_USER_PROFILE	Portlet attributes and metadata.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
USER_TYPE_ID	INT, NOT_NULL			User Type ID.
SUB_NAME	STRING	varchar2(80)		Friendly name of this subscription (bookmark)..
SUB_DESC	STRING	varchar2(255)		Description of this subscription (bookmark).
OBJECT_TYPE	INT, NOT_NULL			Subscribed object type : 0: unclassified 1: channell 2: program 3: category 4: content
SERVICE_ID	INT			Service ID.
CONTENT_TYPE	INT			Content Type.
SYSTEM_ID	INT			System ID.
CHANNEL_ID	INT			Channel ID.
PROGRAM_ID	INT			Program ID.
CATEGORY_ID	INT			Category ID.
CONTENT_ID	INT			Content ID.
DISPLAY_ORDER	INT			Relative order among subscription (bookmark) list.

BV_USER_PROFILE

This table contains contact information attributes and the tax geo code attribute.

Attribute	Type	Column	Friendly name	Semantics
ADDRESS_2	STRING	varchar(255)	Address 2	Additional address line
COMPANY_NAME	STRING	varchar(50)	Company Name	Company Name
MAIL_STOP	STRING	varchar(50)	Mail Stop	Mail Stop
FAX	STRING	varchar(50)	Fax	Fax
TAX_GEO_CODE	STRING	varchar(2)	Geo Code	Geo Code used by Taxware

BV_MR_USER_PROFILE

This table contains purchase history information, shipping and billing addresses.

Attribute	Type	Column	Friendly name	Semantics	Default
MR_DAYS_PURCHASE	LONG		Days since purchase	Days since last purchase	0
MR_PERC_PURCHASE	LONG		Purchase percentage	Percentage of visits that resulted in purchase	0
MR_PURCHASE_INTV	LONG		Average days between purchase	Average days between purchase	0
MR_LAST_PURCHASE	DATETIME		Date of last purchase	Date of last purchase	CURRENT_DATE
MR_NUM_PURCHASE	LONG		Number of purchases	Number of purchases	0
MR_TOT_PURCHASE	MONEY		Total amount purchased	Total amount purchased	0
SHIP_ADDR_PREF	STRING	varchar(50)	Preferred Shipping Address	Alias of Visitor Default Account Shipping Address	
BILL_ADDR_PREF	STRING	varchar(50)	Preferred Billing Address	Alias of Visitor Default Account Billing Address	
SHIP_METH_PREF	STRING	varchar(50)	Preferred Shipping Method	Visitor Default Account Shipping Method	
DEPARTMENT	STRING	varchar(50)	Department	Visitor Department in Company	
FIRST_NAME	STRING	varchar(100)	First Name	First Name	
MIDDLE_NAME	STRING	varchar(55)	Middle Name	Middle Name	
LAST_NAME	STRING	varchar(100)	Last Name	Last Name	
EXT_EMP_ID_REF	STRING	varchar(50)	External Employee ID Ref	External Employee ID Ref	

Attribute	Type	Column	Friendly name	Semantics	Default
DELEGATEE	LONG		Delegatee User Id	Delegatee User Id	
JOB_TITLE	STRING	varchar(50)	Job Title	Job Title	
ACCOUNT_ID_PREF	LONG		Preferred Account ID	Preferred Account ID	0

MR_ADD_BOOK

The MR_ADD_BOOK list table contains Shipping Address Book attributes.

Attribute	Type	Column	Friendly name	Semantics
AB_ALIAS **ATTR_KIND:** KEY, REQ_ATTR	STRING	varchar(50)	Shipping Alias	Shipping Alias
AB_NAME	STRING	varchar(50)	Shipping Name	Name
AB_ADDRESS	STRING	varchar(255)	Shipping Address	Address
AB_ADDRESS_2	STRING	varchar(255)	Shipping Address 2	Additional Shipping Address line
AB_COMPANY_NAME	STRING	varchar(50)	Shipping Company Name	Shipping Company Name
AB_CITY	STRING	varchar(50)	Shipping City	City
AB_STATE	STRING	varchar(20)	Shipping State	State
AB_ZIP	STRING	varchar(20)	Shipping Zip Code	Zip Code
AB_COUNTRY	STRING	varchar(20)	Shipping Country	Country
AB_EMAIL	STRING	varchar(50)	Shipping E-mail address	E-mail address
AB_PHONE	STRING	varchar(50)	Shipping Phone	Phone
AB_FAX	STRING	varchar(50)	Shipping Fax	Fax
AB_SHIPPINGTYPE	STRING	varchar(50)	Shipping Method	Shipping Method
AB_MAIL_STOP	STRING	varchar(50)	Shipping Mail Stop	Mail Stop
AB_DEPARTMENT	STRING	varchar(50)	Shipping Department	Department
AB_TAX_GEO_CODE	STRING	varchar(2)	Geo Code	Geo Code used by Taxware

MR_INTERESTS

The MR_INTERESTS list table contains Product Interests attributes.

Attribute	Type	Column	Friendly name	Semantics
INT_AREA **ATTR_KIND:** KEY, REQ_ATTR	MR_INTEREST_TYPE	varchar(50)	Name of Interest Area	Name of Interest Area
INT_LEVEL	LONG		Interest Level	Numerical Level of Interests

MR_PURCHASED_ITEMS

The MR_PURCHASED_ITEMS list table is for maintaining the list of products (Purchased Items) the visitor bought.

Attribute	Type	Column	Friendly name	Semantics
PI_STORE_ID	LONG		Store ID	Store ID
PI_ORDER_DATE **ATTR_KIND:** KEY, REQ_ATTR	DATETIME		Date of Purchase	Date/Time of Purchase
PI_ITEM_NO **ATTR_KIND:** KEY, REQ_ATTR	LONG		Item Number	Item Number
PI_CONTENT_TYPE	LONG		Content Type	Content Type of Item
PI_PRODUCT_ID	STRING	varchar(80)	Product ID	Product ID of Item
PI_DEPARTMENT	MR_PROD_DEPT_TYPE	varchar(30)	Department	Product Department
PI_QUANTITY	LONG		Quantity	Item Quantity
PI_TOTAL_PRICE	MONEY		Total Price	Total Price Paid for Item

MR_ORDER_HISTORY

The MR_ORDER_HISTORY list table maintains the order history summary of a visitor.

Attribute	Type	Column	Friendly name	Semantics
OH_STORE_ID **ATTR_KIND:** KEY, REQ_ATTR	LONG		Store ID	Store ID
OH_ORDER_DATE **ATTR_KIND:** KEY, REQ_ATTR	DATETIME		Date of Order	Date/Time of Order
OH_ORDER_NUMBER	STRING	VARCHAR(50)	Order Number	Order Number
OH_TOTAL_DISCNT	MONEY		Total Discount	Total Discount
OH_GRAND_TOTAL	MONEY		Total Price	Total Price Paid

MR_PERSISTENT_CART

The MR_PERSISTENT_CART list table maintains the persistent shopping cart of a visitor.

Attribute	Type	Friendly name	Semantics
PC_STORE_ID **ATTR_KIND:** KEY, REQ_ATTR	LONG	Store ID	Store ID
PC_ITEM_NO **ATTR_KIND:** KEY, REQ_ATTR	LONG	Item Number	Item Number
PC_CONTENT_TYPE	LONG	Content Type	Content Type
PC_CONTENT_OID	LONG	Content OID	Content OID
PC_QUANTITY	LONG	Quantity	Quantity

MR_DELEGATEES

The MR_DELEGATEES list table contains delegatee attributes for a user.

Attribute	Type	Friendly name	Semantics
DG_USER_ID **ATTR_KIND:** KEY, REQ_ATTR	LONG	Delegatee User ID	Delegatee User ID

BV_MR_USER_ACCT

The BV_MR_USER_ACCT list table contains account attributes for a user.

Attribute	Type	Column	Friendly name	Semantics
MRACCOUNT_ID **ATTR_KIND:** KEY, REQ_ATTR	LONG		Account ID	Account ID
MRSHIP_ADDR_PREF	STRING	varchar(50)	Preferred Shipping Address	Alias of Visitor Default Account Shipping Address
MRBILL_ADDR_PREF	STRING	varchar(50)	Preferred Billing Address	Alias of Visitor Default Account Billing Address
MRSHIP_METH_PREF	STRING	varchar(50)	Preferred Shipping Method	Visitor Default Account Shipping Method
MRDEPARTMENT	STRING	varchar(50)	Department	Visitor Department in Company
MRPMT_PREF	STRING	varchar(50)	Preferred Payment Method	Alias of Visitor Default Account Payment Method

BV_MR_ACCT_ROLE

The BV_MR_ACCT_ROLE list table contains visitor account role attributes.

Attribute	Type	Friendly name	Semantics
ACCOUNT_ID **ATTR_KIND:** KEY, REQ_ATTR	LONG	Account ID	Account ID applied to visitor role
ACCOUNT_ROLE **ATTR_KIND:** KEY, REQ_ATTR	ROLE	Account Role	Role that the visitor will assume
CURR_FLAG	LONG	Current Role	Flag to indicate whether the role is current

MR_PC_ITEM_PROPS

The MR_PC_ITEM_PROPS list table is a child table of the MR_PERSISTENT_CART table and contains persistent shopping cart item property attributes.

Attribute	Type	Column	Friendly name	Semantics
PCI_STORE_ID **ATTR_KIND:** KEY, REQ_ATTR	LONG		Item - Store ID	Item - Store ID
PCI_ITEM_NO **ATTR_KIND:** KEY, REQ_ATTR	LONG		Item - Item Number	Item - Item Number
PCI_ITEM_PROP_NAME **ATTR_KIND:** KEY, REQ_ATTR	STRING NOT NULL	VARCHAR(50)	Item - Property Name	Item - Property Name
PCI_ITEM_PROP_VALUE	STRING	VARCHAR(255)	Item - Property Value	Item - Property Value

MR_USER_ORG

The MR_USER_ORG list table contains user organization attributes.

Attribute	Type	Column	Friendly name	Semantics
ORG_CODE **ATTR_KIND:** KEY, REQ_ATTR	STRING	varchar(32)	Organization Code	Organization Code
ORG_OID	LONG		Organization OID	Organization OID

MR_TAX_UPROF

The MR_TAX_UPROF list table contains sales / use tax exemption data attributes for a user.

Attribute	Type	Column	Friendly name	Semantics
STEP_CUST_ID **ATTR_KIND:** KEY, REQ_ATTR	STRING	varchar(20)	STEP Customer ID	STEP Name
MERCHANT_NAME **ATTR_KIND:** KEY	STRING	varchar(20)	Merchant/Company ID	Merchant Name
REASON_CODE **ATTR_KIND:** KEY	STRING	varchar(2)	Tax Exemption Reason Code	Reason_Code

Category content type tables

These tables contain extensions to the CATEGORY_CONTENT content type used for organization entities.

- "BV_CATEGORY" on page 307
- "MR_PS_ATTRIBUTES" on page 308
- "MR_ORG_USERS" on page 308

- "MR_ORG_EXP_CODES" on page 308
- "MR_CAT_FEATURES" on page 308
- "MR_ORG_ACCT_CODE" on page 309

BV_CATEGORY

The `category_content_ext_cwa.src` file contains extensions to the BV_CATEGORY table in One-To-One Commerce. These extensions to the BV_CATEGORY table include product comparison attributes, extended information attributes, organization entity information attributes, and type tree information.

Attribute	Type	Column	Friendly name	Semantics	Default
MR_PCM_ATTRIBUTES	STRING	varchar(255)	PCM Attributes	List of product attributes	
MR_LONG_DESC	STRING	varchar(255)	Long Description	Category Long Description	
MR_SHORT_DESC	STRING	varchar(128)	Short Description	Category Short Description	
MR_AUX	STRING	varchar(128)	Auxiliary Column	Category Auxiliary Column	
ACCOUNT_ID	LONG		Account ID	Account ID	
OET_NAME	STRING	varchar(50)	Organization Entity Type Name	Organization Entity Type Name	
INTERNAL_CODE	STRING	varchar(255)	Organization Code	Organization Code	
APPROVER_ID	LONG		Approver ID	Organization Entity Approver ID	
APPROVAL_LIMIT	MONEY		Approval Limit	Organization Entity Approval Limit	
REQUISITION_LIMIT	MONEY		Requisition Limit	Organization Entity Requisition Limit	
REQUISITION_STOP	MONEY		Requisition Stop	Organization Entity Requisition Stop	
OVERRIDE_LIMIT	BOOLEAN_ENUM		Override Limit Flag	Organization Entity Override Limit Flag	
MR_UNSPSC	STRING	varchar(12)	UNSPSC product category	UNSPSC product category	
MR_TYPE_TREE	LONG		Type Tree Indicator	Value 1 indicates the category is a type tree	0

MR_PS_ATTRIBUTES

The MR_PS_ATTRIBUTES list table contains search attributes.

Attribute	Type	Column	Friendly name	Semantics
PS_ATTR_NAME **ATTR_KIND:** KEY, REQ_ATTR	STRING	varchar(250)	Friendly Name	Friendly Name for Parametric Search Attribute
PS_PRECEDENCE **ATTR_KIND:** KEY, REQ_ATTR	LONG		Precedence	Precedence for Parametric Search Attribute

MR_ORG_USERS

The MR_ORG_USERS list table contains organization entity members.

Attribute	Type	Friendly name	Semantics
USER_ID **ATTR_KIND:** KEY, REQ_ATTR	LONG	User ID	User ID for Member of Organization Entity

MR_ORG_EXP_CODES

The MR_ORG_EXP_CODES list table contains organization entity expense codes.

Attribute	Type	Column	Friendly name	Semantics
EXP_CHARGE_CODE **ATTR_KIND:** KEY, REQ_ATTR	STRING	VARCHAR(30)	Expense Charge Code	Expense Charge Code

MR_CAT_FEATURES

The MR_CAT_FEATURES list table contains category features.

Attribute	Type	Friendly name	Semantics	Default
FEATURE_TYPE_ID **ATTR_KIND:** KEY, REQ_ATTR	OID_FEATURE_TYPE NOT NULL	Feature OID	Feature OID	
FEATURE_INHERITED	LONG	Feature Inherited Indicator	Value 1 indicates the feature is inherited	0
FEATURE_SEARCHABLE	LONG	Feature Searchable Indicator	Value 1 indicates the feature is searchable	1

MR_ORG_ACCT_CODE

The MR_ORG_ACCT_CODE list table contains default accounting code attributes for an organization entity.

Attribute	Type	Friendly name	Semantics
COA_KEY **ATTR_KIND:** KEY, REQ_ATTR	LONG	The COA key from the MR_COA_CODES table	Also used as a key since the COA_CODE_KEY is unique

Account profile content type tables

These tables contain extensions to the MR_ACCOUNT_PROFILE content type used by the One-To-One Commerce application.

MR_ACCT_PROFILE

Contains account profile content type attributes.

Schema file	mr_account_profile_base.src, mr_account_profile_ext.src	
Table	MR_ACCT_PROFILE	This name is required; do not change it
Class	MR_ACCOUNT_PROFILE	This table is hidden; it may not be viewed directly from the Command Center
Service	ALL_STORE	This table is available to the entire site
Content	MR_ACCT_PROFILE "Account Profile"	Type 301

All of the following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
ACCOUNT_NAME	STRING NOT_NULL	varchar(50)	REQ_ATTR	Account Name
STORE_ID	LONG NOT_NULL	—	REQ_ATTR, RONLY	Interactive Service ID (Read-only)
CREATION_TIME	DATETIME NOT_NULL	—	REQ_ATTR, RONLY	Date/Time Payment Type was Created (Read-only)
STATUS	STATUS_TYPE NOT_NULL	—	REQ_ATTR	Available/Unavailable for Display on Web Site
DELETED	LONG NOT_NULL	—	REQ_ATTR, RONLY, HIDDEN	Remove From/Keep in Database
LAST_MOD_TIME	DATETIME NOT_NULL	—	REQ_ATTR, RONLY	Date/Time of Last Modification to Product (Read-only)
ACCOUNT_ID	LONG NOT_NULL	—	REQ_ATTR	Account ID
Extensions (mr_account_profile_ext.src)				
COMPANY_NAME	STRING	varchar(50)		Company Name
INDUSTRY	MR_INDUSTRY_TYPE			Company Industry
CONTACT	STRING	varchar(50)		Contact
ADDRESS	STRING	varchar(255)		Account Address
ADDRESS_2	STRING	varchar(255)		Account Address 2
MAIL_STOP	STRING	varchar(50)		Mail Stop
CITY	STRING	varchar(50)		Account Address - City
STATE	STRING	varchar(20)		Account Address - State
ZIP	STRING	varchar(20)		Account Address - Zip
COUNTRY	STRING	varchar(20)		Account Address - Country
PHONE	STRING	varchar(50)		Phone
FAX	STRING	varchar(50)		Fax
EMAIL	STRING	varchar(50)		Email
ACCOUNT_TYPE	MR_ACCOUNT_TYPE			Account Type
SRV_STORE_ID	LONG			Service Store ID
CONTACT_ID	LONG			Contact User ID

Name	Type	Column	Attribute Kind	Friendly name or Semantics
ACTIVE	BOOLEAN_ENUM			Active
REGION	MR_REGION_TYPE			Region
COMPANY_RATING	MR_CO_RATING_TYPE			Company Rating
COMPANY_DESC	TEXT			Company Description
COMPANY_LOGO	STRING	varchar(255)		Company Logo
COMPANY_URL	STRING	varchar(100)		Company URL
NUM_EMPL	LONG			Number of Employees
REVENUE	MONEY			Revenue
TOT_NUM_TRANS	LONG			Number of Trans
TOT_NUM_LOGIN	LONG			Number of Login
DEF_CONTRACT_ID	LONG			Default Contract ID
LIFE_CYCLE_STATE	MR_LIFECYCLE_TYPE			Life Cycle State
CATALOG_FLAG	BOOLEAN_ENUM			Indicates if this supplier's catalog should be viewed on the supplier's site or locally
TAX_ID	STRING	varchar(255)		Tax identifier
TOP_CAT_OID	LONG			Account Root Catalog OID
BROWSE_CAT_OID	LONG			Start Browse Catalog OID
HOLD_PO_GEN	BOOLEAN_ENUM			Hold PO Generation

MR_DEPARTMENTS

The `mr_account_profile_ext.src` file contains the MR_DEPARTMENTS list table. The MR_DEPARTMENTS list table contains department attributes for an account.

Attribute	Type	Column	Friendly name	Semantics
DEPARTMENT **ATTR_KIND:** KEY, REQ_ATTR	STRING NOT_NULL	varchar(50)	Department	Department

MR_SHIP_ADDR

The `mr_account_profile_ext.src` file contains the MR_SHIP_ADDR list table. The MR_SHIP_ADDR list table contains shipping address attributes for an account.

Attribute	Type	Column	Friendly name	Semantics
SHIP_ALIAS **ATTR_KIND:** KEY, REQ_ATTR	STRING NOT_NULL	varchar(50)	Shipping Alias	Shipping Alias
SHIP_NAME	STRING	varchar(50)	Shipping Name	Shipping Name
SHIP_ADDRESS	STRING	varchar(255)	Shipping Address	Shipping Address
SHIP_ADDRESS_2	STRING	varchar(255)	Shipping Address 2	Shipping Address 2

Attribute	Type	Column	Friendly name	Semantics
SHIP_MAIL_STOP	STRING	varchar(50)	Shipping Address - Mail Stop	Shipping Address - Mail Stop
SHIP_CITY	STRING	varchar(50)	Shipping Address - City	Shipping Address - City
SHIP_STATE	STRING	varchar(20)	Shipping Address - State	Shipping Address - State
SHIP_ZIP	STRING	varchar(20)	Shipping Address - Zip	Shipping Address - Zip
SHIP_COUNTRY	STRING	varchar(20)	Shipping Address - Country	Shipping Address - Country
SHIP_DEPARTMENT	STRING	varchar(50)	Shipping Address - Department	Shipping Address - Department
SHIP_PHONE	STRING	varchar(50)	Shipping Address - Phone	Shipping Address - Phone
SHIP_FAX	STRING	varchar(50)	Shipping Address - Fax	Shipping Address - Fax
SHIP_EMAIL	STRING	varchar(50)	Shipping Address - Email	Shipping Address - Email
TAX_GEO_CODE	STRING	varchar(2)	Geo Code	Geo Code used by Taxware

MR_BILL_ADDR

The `mr_account_profile_ext.src` file contains the MR_BILL_ADDR list table. The MR_BILL_ADDR list table contains billing address attributes for an account.

Attribute	Type	Column	Friendly name	Semantics
BILL_ALIAS **ATTR_KIND:** KEY, REQ_ATTR	STRING NOT_NULL	varchar(50)	Billing Alias	Billing Alias
BILL_NAME	STRING	varchar(50)	Billing Name	Billing Name
BILL_ADDRESS	STRING	varchar(255)	Billing Address	Billing Address
BILL_ADDRESS_2	STRING	varchar(255)	Billing Address 2	Billing Address 2
BILL_MAIL_STOP	STRING	varchar(50)	Billing Address - Mail Stop	Billing Address - Mail Stop
BILL_CITY	STRING	varchar(50)	Billing Address - City	Billing Address - City
BILL_STATE	STRING	varchar(20)	Billing Address - State	Billing Address - State
BILL_ZIP	STRING	varchar(20)	Billing Address - Zip	Billing Address - Zip
BILL_COUNTRY	STRING	varchar(20)	Billing Address - Country	Billing Address - Country
BILL_DEPARTMENT	STRING	varchar(50)	Billing Address - Department	Billing Address - Department
BILL_PHONE	STRING	varchar(50)	Billing Address - Phone	Billing Address - Phone
BILL_FAX	STRING	varchar(50)	Billing Address - Fax	Billing Address - Fax
BILL_EMAIL	STRING	varchar(50)	Billing Address - Email	Billing Address - Email

MR_SHIP_METHODS

The `mr_account_profile_ext.src` file contains the MR_SHIP_METHODS list table. The MR_SHIP_METHODS list table contains shipping methods attributes for an account.

Attribute	Type	Column	Friendly name	Semantics
SHIP_METHOD **ATTR_KIND:** KEY, REQ_ATTR	STRING NOT_NULL	varchar(50)	Shipping Method	Shipping Name

MR_ACCOUNT_REPS

The `mr_account_profile_ext.src` file contains the MR_ACCOUNT_REPS list table. The MR_ACCOUNT_REPS list table contains representative attributes for an account.

Attribute	Type	Column	Friendly name	Semantics
USER_ID **ATTR_KIND:** KEY, REQ_ATTR	LONG NOT_NULL	varchar(50)	Account Rep User ID	Account Rep User ID

MR_EXT_PARTY_ID

The `mr_account_profile_ext.src` file contains the MR_EXT_PARTY_ID list table. The MR_EXT_PARTY_ID list table contains external party ID reference attributes for an account.

Attribute	Type	Column	Friendly name	Semantics
EP_KEY **ATTR_KIND:** KEY, REQ_ATTR	STRING	varchar(100)	External Party ID Key (Agency ID + Party ID)	External Party ID Key (Agency ID + Party ID)
EP_AGENCY_ID **ATTR_KIND:** REQ_ATTR	STRING	varchar(50)	External Party ID Reference Agency ID	External Party ID Reference Agency ID
EP_PARTY_ID **ATTR_KIND:** REQ_ATTR	STRING	varchar(50)	External Party ID Reference Party ID	External Party ID Reference Party ID
ACCOUNT_ID	LONG		Account ID of the account that this ID is specific to	Account ID of the account that this ID is specific to (for example, a buyer's account number with a supplier)
DESCRIPTION	STRING	varchar(255)	Description of this identifier	Description of this identifier

MR_OE_TYPES

The `mr_account_profile_ext.src` file contains the MR_OE_TYPES list table. The MR_OE_TYPES list table contains organization entity type attributes.

Attribute	Type	Column	Friendly name	Semantics
OET_NAME **ATTR_KIND:** KEY, REQ_ATTR	STRING	varchar(50)	Organization Entity Type Name	Organization Entity Type Name
OET_APPR_LIMIT	MONEY		Organization Entity Type Approval Limit	Organization Entity Type Approval Limit
OET_REQ_LIMIT	MONEY		Organization Entity Type Requisition Limit	Organization Entity Type Requisition Limit
OET_REQ_STOP	MONEY		Organization Entity Type Requisition Stop	Organization Entity Type Requisition Stop

MR_ACCT_TRANS

The `mr_account_profile_ext.src` file contains the MR_ACCT_TRANS list table. The MR_ACCT_TRANS list table contains transportation attributes.

Attribute	Type	Column	Friendly name	Semantics
TRANS_ALIAS **ATTR_KIND:** KEY, REQ_ATTR	STRING	varchar(50)	Transportation Alias	Transportation Alias
TRANS_MODE	STRING	varchar(70)	Transportation Mode	Transportation Mode
TRANS_MEAN	STRING	varchar(70)	Transportation Mean	Transportation Mean
TRANS_CARRIER	STRING	varchar(70)	Transportation Carrier	Transportation Carrier
TRANS_CSCN	STRING	varchar(35)	Transportation Customer Shipping Contract Number	Transportation Customer Shipping Contract Number
TRANS_SHIP_INSTR	STRING	varchar(255)	Transportation Shipping Instructions	Transportation Shipping Instructions

MR_PMT_ACCEPT

The `mr_account_profile_ext.src` file contains the MR_PMT_ACCEPT list table. The MR_PMT_ACCEPT list table contains accepted payment types.

Attribute	Type	Friendly name	Semantics
PA_PMTTYPE_ID **ATTR_KIND:** KEY, REQ_ATTR	LONG NOT_NULL	Payment Type ID	Payment Type ID Semantics

MR_PMT_TERMS

The `mr_account_profile_ext.src` file contains the MR_PMT_TERMS list table. The MR_PMT_TERMS list table contains payment terms attributes.

Attribute	Type	Column	Friendly name	Semantics
PT_TERM_CODE **ATTR_KIND:** KEY, REQ_ATTR	STRING NOT_NULL	varchar(50)	Term Code	Payment Term Code
PT_PMTTYPE_ID	LONG NOT_NULL		Payment Type ID	Payment Type ID Semantics
PT_DISC_PERCENT	STRING	varchar(10)	Discount	Payment Percentage Discount
PT_DISC_DAYS_DUE	LONG		Discount Days Due	Discount Days Due
PT_DISC_TIME_REF	STRING	varchar(50)	Discount Time Reference	Discount Time Reference
PT_DISC_DUE_DATE	DATETIME		Discount Due Date	Discount Due Date
PT_NET_DAYS_DUE	LONG		Net Days Due	Net Days Due
PT_NET_TIME_REF	STRING	varchar(50)	Net Time Reference	Net Time Reference
PT_EXTRA_INFO	TEXT		Additional Info	Additional Info

MR_PROD_AREAS

The `mr_account_profile_ext.src` file contains the MR_PROD_AREAS list table. The MR_PROD_AREAS list table contains product interest areas.

Attribute	Type	Column	Friendly name	Semantics
PR_ACCT_TYPE **ATTR_KIND:** KEY, REQ_ATTR	MR_ACCOUNT_TYPE		Interest by Account Type	Interest by Account Type
PR_PROD_AREA **ATTR_KIND:** KEY, REQ_ATTR	STRING NOT_NULL	varchar(50)	Product Interest Area	Product Interest Area

MR_CREQ_HEADERS

The `mr_account_profile_ext.src` file contains the MR_CREQ_HEADERS list table. The MR_CREQ_HEADERS list table contains headers for insertion into the catalog request setup.

Attribute	Type	Column	Friendly name	Semantics
CREQ_INDEX **ATTR_KIND:** KEY, REQ_ATTR	LONG NOT_NULL		The Index key number	The Index key number
HDR_TYPE **ATTR_KIND:** REQ_ATTR	HDR_TYPE_ENUM		Type of header the supplier needs in a setup request	From or Sender

Attribute	Type	Column	Friendly name	Semantics
AGENCY_ID **ATTR_KIND:** REQ_ATTR	STRING NOT_NULL	varchar(50)	Domain of the credential (for example DUNS). Lookup to MR_EXT_PARTY_ID table	Domain of the credential (for example DUNS). Lookup to MR_EXT_PARTY_ID table
SUPP_ACCT_ID **ATTR_KIND:** REQ_ATTR	LONG NOT_NULL		Account ID of the supplier	Account ID of the supplier
PASSWORD	STRING	varchar(50)	Password	Password between buyer and supplier

MR_XML_FORMAT

The `mr_account_profile_ext.src` file contains the MR_XML_FORMAT list table. The MR_XML_FORMAT list table contains XML format information for a supplier or marketplace.

Attribute	Type	Column	Friendly name	Semantics
TRANS_TYPE **ATTR_KIND:** KEY, REQ_ATTR	TRANS_TYPE_ENUM		Document type (for example: Catalog Request, Purchase Order)	Document type (for example: Catalog Request, Purchase Order)
XML_STANDARD **ATTR_KIND:** REQ_ATTR	XML_TYPE_ENUM		XML standard that this document type conforms to	XML standard that this document type conforms to
URL	STRING	varchar(200)	URL to send this document type to	URL to send this document type to
POST_BACK_URL	STRING	varchar(200)	Script path or URL to handle the shopping cart (or other “reply” documents)	Script path or URL to handle the shopping cart (or other “reply” documents)
POST_PROTOCOL	POST_PROT_ENUM		Protocol to use when posting this document type	Protocol to use when posting this document type
ATTACHMENTS	BOOLEAN_ENUM		Whether the supplier accepts attachments with this document type	Whether the supplier accepts attachments with this document type

MR_TAX_APROF

The `mr_account_profile_ext.src` file contains the MR_TAX_APROF list table. The MR_TAX_APROF list table contains Sales / Use Tax Exemption data for a user.

Attribute	Type	Column	Friendly name	Semantics
STEP_CUST_ID **ATTR_KIND:** KEY, REQ_ATTR	STRING	varchar(20)	STEP Customer ID	STEP Name
MERCHANT_NAME **ATTR_KIND:** KEY	STRING	varchar(20)	Merchant/Company ID	Merchant Name
REASON_CODE **ATTR_KIND:** KEY	STRING	varchar(2)	Tax Exemption Reason Code	Reason_Code

MR_PARTNER_NETWORK

The `mr_account_profile_ext.src` file contains the MR_PARTNER_NETWORK list table. The MR_PARTNER_NETWORK list table contains partner network reference attributes.

Attribute	Type	Column	Friendly name	Semantics
NETWORK_ACCT_ID **ATTR_KIND:** KEY, REQ_ATTR	LONG		Network Account ID	Network Account ID
PARTNER_USER_ID	LONG NOT_NULL		Trading Partner Default's User ID	Trading Partner Default's User ID
POSTBACK_URL	STRING	varchar(255)	Postback URL	Postback URL

MR_SUPPLIER_LIST

The `mr_account_profile_ext.src` file contains the MR_SUPPLIER_LIST list table. The MR_SUPPLIER_LIST list table contains a list of suppliers for an account.

Attribute	Type	Friendly name	Semantics
SUPPLIER_ID **ATTR_KIND:** KEY, REQ_ATTR	LONG NOT_NULL	Supplier Acct ID	Supplier Account ID
SUPPLIER_OID **ATTR_KIND:** REQ_ATTR	LONG NOT_NULL	Supplier OID	Supplier OID

Qualifier tables

Qualifiers filter the content in an InfoExchange Portal site. The qualifier value name is unique within a qualifier, but may not be unique among all qualifiers. Each qualifier must have at least one value.

BV_EP_QUAL_VALUE

Defines the qualifier values for the InfoExchange Portal site.

Schema file	`ep_qual_val_base.src`	
Table	BV_EP_QUAL_VALUE	This name is required; do not change it.
Class	EP_QUAL_VALUE	This table is hidden; it may not be viewed directly from the Command Center
Service	ALL_STORE	This table is available to the entire site
Content	EP_QUAL_VALUE “InfoExchange Qualifier Value”	Type 315

All the following attributes are required. You may add additional attributes, but you may not change the attributes listed in the following table.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
“Basics” group				
QV_NAME	STRING NOT_NULL	varchar(162)	REQ_ATTR	Qualifier Value Key.
STORE_ID	LONG NOT_NULL	—	REQ_ATTR, RONLY, HIDDEN	Interactive Service ID (Read-only).
CREATION_TIME	DATETIME NOT_NULL	—	REQ_ATTR, RONLY	Date/Time Qualifier Value was Created (Read-only)
STATUS	STATUS_TYPE NOT_NULL	—	REQ_ATTR	Available/Unavailable for Display on Web Site.
DELETED	LONG NOT_NULL	—	REQ_ATTR, RONLY, HIDDEN	Remove From/Keep in Database.
LAST_MOD_TIME	DATETIME NOT_NULL	—	REQ_ATTR, RONLY	Date/Time of Last Modification to Qualifier Value (Read-only).
“Details” group				
FRIENDLY_NAME	STRING NOT_NULL	varchar(80)	REQ_ATTR	Qualifier Value Name.
USER_SETTABLE	BOOLEAN_ENUM NOT_NULL	—	REQ_ATTR	Is the Qualifier Value Settable By the Visitor?

InfoExchange Portal 6.0.0 supports qualifiers on content. By default, InfoExchange Portal setup creates qualifier list tables for two content types: PRODUCT and EDITORIAL. The schema files for these tables are generated dynamically by the InfoExchange Portal setup script during the setup process. The script used to generate the schema files is `/projects/knowledge/root/samples/dbschema/add_qv_list_table`.

BV_EP_QV0

This is a list table on the 'PRODUCT' content table.

Table	BV_EP_QV0	This name is required; do not change it.
Related to	BV_PRODUCT	The qualifier value settings on product.

All of the following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
QID	LONG, NOT_NULL		KEY, HIDDEN, REQ_ATTR	Qualifier ID
QVID	LONG, NOT_NULL		KEY, REQ_ATTR	Qualifier Value ID

BV_EP_QV2

This is a list table on the 'EDITORIAL' content table.

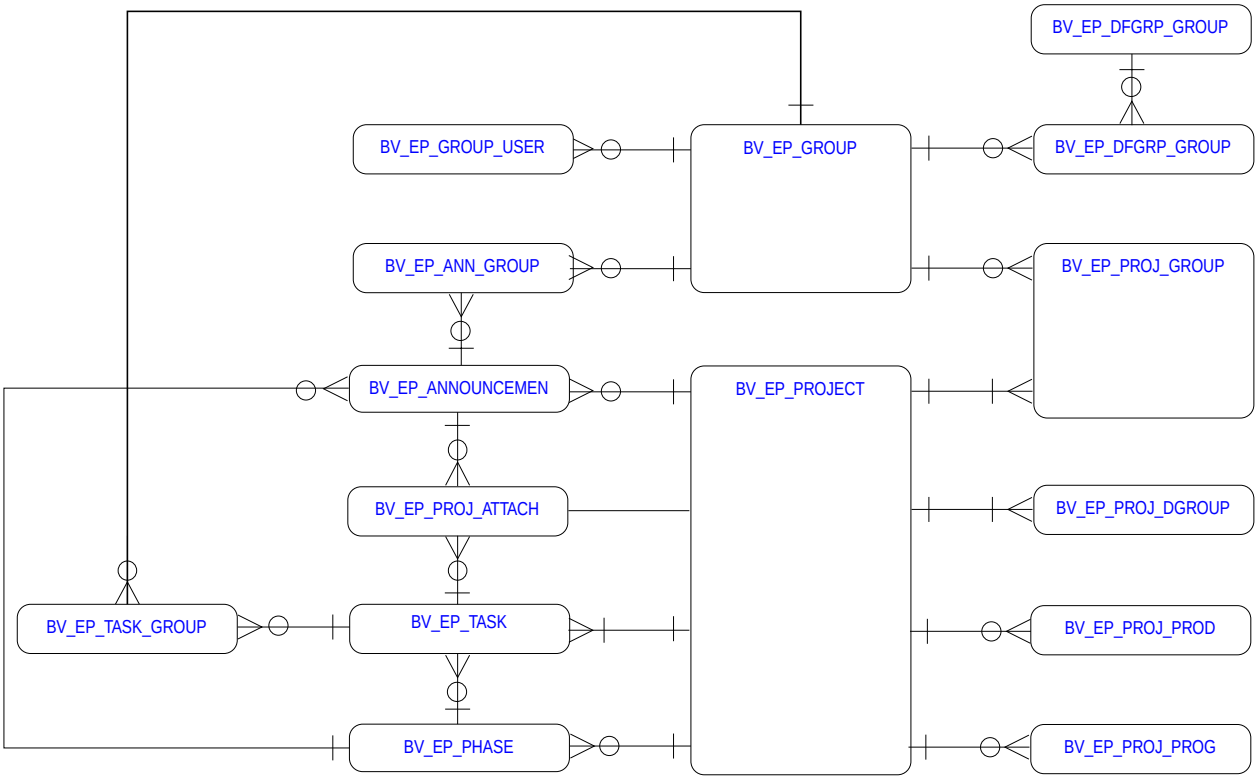
Table	BV_EP_QV2	This name is required; do not change it.
Related to	BV_EDITORIAL	The qualifier value settings on EDITORIAL.

All of the following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
QID	LONG, NOT_NULL		KEY, HIDDEN, REQ_ATTR	Qualifier ID
QVID	LONG, NOT_NULL		KEY, REQ_ATTR	Qualifier Value ID

Project tables

The project tables contain information about projects.



BV_EP_PROJECT

Lists the project attributes.

Schema file	`ep_project_base.src`, `ep_project_ext.src`	
Table	BV_EP_PROJECT	This name is required; do not change it.
Child	BV_EP_PROJ_DGROUP	Discussion groups.
Child	BV_EP_PROJ_PROD	Products.
Child	BV_EP_PROJ_PROG	Programs.
Child	BV_EP_PROJ_ATTACH	Attachments.
Child	BV_EP_PROJ_GROUP	Groups.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
“Basics” group (`ep_project_base.src`)				
PROJECT_NAME	STRING, NOT_NULL	varchar (80)	KEY, REQ_ATTR	Project name
STORE_ID	LONG, NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Service ID
CREATION_TIME	DATETIME, NOT_NULL	-	REQ_ATTR, RONLY	Creation time
STATUS	STATUS_TYPE NOT_NULL	-	REQ_ATTR	On-line status
DELETED	LONG NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Deleted status
LAST_MOD_TIME	DATETIME NOT_NULL	-	REQ_ATTR, RONLY	Last modified time
“Details” group (`ep_project_ext.src`)				
PROJECT_DESC	TEXT	-	-	Description
PROJECT_GOAL	TEXT	-	-	Goal
PROJECT_PAGE_PATH	STRING	varchar (255)	-	Path to script file relative to the script root. Not used in InfoExchange Portal 6.0.0.
PROJECT_ICON_PATH	FILE_PATH	varchar (255)	-	Path to icon image file relative to the HTTP document root.
CURRENT_PHASE	OID_PHASE	-	-	Current phase
STRCOL1	STRING	varchar (255)	-	Additional text 1, for customer use.
STRCOL2	STRING	varchar (255)	-	Additional text 2, for customer use.
STRCOL3	STRING	varchar (255)	-	Additional text 3, for customer use.

BV_EP_PROJ_DGROUP

Lists the discussion groups in the project.

Schema file	ep_project_ext.src	
Table	BV_EP_PROJ_DGROUP	This name is required; do not change it.
Parent	BV_EP_PROJECT	Project table.
Related to	BV_DF_GRP	Discussion group for the project.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
DGROUP_NAME	TEXT, NOT_NULL	varchar (250)	KEY, REQ_ATTR	Discussion group name.
PERMISSION	ACCESS_TYPE NOT_NULL	-	REQ_ATTR	Project group in which members are allowed to see and post messages in this discussion group (OBSOLETE).
DF_GRP_OID	LONG	-	REQ_ATTR	Discussion group OID

BV_EP_PROJ_PROD

Lists the products in the project.

Schema file	ep_project_ext.src	
Table	BV_EP_PROJ_PROD	This name is required; do not change it.
Parent	BV_EP_PROJECT	Project table

Name	Type	Column	Attribute Kind	Friendly name or Semantics
PRODUCT_OID	OID_PRODUCT NOT_NULL	-	KEY, REQ_ATTR	Product OID (BV_PRODUCT).

BV_EP_PROJ_PROG

Lists the programs in the project.

Schema file	ep_project_ext.src	
Table	BV_EP_PROJ_PROG	This name is required; do not change it.
Parent	BV_EP_PROJECT	Project table

Name	Type	Column	Attribute Kind	Friendly name or Semantics
PROGRAM_OID	OID_PROGRAM NOT_NULL	-	KEY, REQ_ATTR	Program ID (BV_KM_PROGRAM).
PHASE_OID	OID_PHASE NOT_NULL	-	KEY, REQ_ATTR	Phase ID of the phase in which the program is displayed (BV_EP_PHASE). The special value 0 means the program is visible in all phases.

BV_EP_PROJ_ATTACH

Lists the project attachments.

Schema file	ep_project_ext.src	
Table	BV_EP_PROJ_ATTACH	This name is required; do not change it.
Parent	BV_EP_PROJECT	Project table.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
ATTACHMENT_NAME	STRING, NOT_NULL	varchar (80)	KEY, REQ_ATTR	label to display for attachment download link.
FILE_DESC	TEXT	-	-	File description.
FILE_PATH	FILE_PATH, NOT_NULL	varchar (255)	REQ_ATTR	File path relative to the HTTP document root where the file was uploaded.
FILE_SIZE	LONG	-	-	File size in bytes.
USER_ID	LONG, NOT_NULL	-	REQ_ATTR	ID of the user who last updated the file.
ASSOC_CONTENT	LONG, NOT_NULL	-	KEY, REQ_ATTR	Content OID of the task, meeting, or announcement to which the file was attached.
ASSOC_CONTENT_TYPE	LONG, NOT_NULL	-	KEY, REQ_ATTR	Content type of the task, meeting, or announcement to which the file was attached.

BV_EP_PROJ_GROUP

Lists the project attachments.

Schema file	`ep_project_ext.src`	
Table	BV_EP_PROJ_GROUP	This name is required; do not change it.
Parent	BV_EP_PROJECT	Project table.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
GROUP_OID	OID_GROUP	-	KEY, REQ_ATTR	Project Group ID.

BV_EP_PHASE

Lists the phase attributes in the project.

Schema file	`ep_phase_base.src`	
Table	BV_EP_PHASE	This name is required; do not change it.
Related to	BV_EP_PROJECT	Projects.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
“Basics” group				
PHASE_NAME	STRING, NOT_NULL	varchar (80)	KEY, REQ_ATTR	Phase name
STORE_ID	LONG, NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Service ID
CREATION_TIME	DATETIME, NOT_NULL	-	REQ_ATTR, RONLY	Creation time
STATUS	STATUS_TYPE NOT_NULL	-	REQ_ATTR	Status
DELETED	LONG NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Deleted status
LAST_MOD_TIME	DATETIME NOT_NULL	-	REQ_ATTR, RONLY	Last modified time
“Details” group				
PHASE_DESC	TEXT	-	-	Description
PHASE_GOAL	TEXT	-	-	Goal
PROJECT_OID	OID_PROJECT NOT_NULL	-	-	Project OID of the project containing the phase (BV_EP_PROJECT).

Name	Type	Column	Attribute Kind	Friendly name or Semantics
STRCOL1	STRING	varchar (255)	-	Additional text 1, for customer use.
STRCOL2	STRING	varchar (255)	-	Additional text 2, for customer use.
STRCOL3	STRING	varchar (255)	-	Additional text 3, for customer use.

BV_EP_TASK

Lists the tasks and meetings in the project.

Schema file	`ep_task_base.src`	
Table	BV_EP_TASK	This name is required; do not change it.
Related to	BV_EP_PROJECT BV_EP_PHASE	Project and project phase tables.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
“Basics” group				
TASK_NAME	STRING, NOT_NULL	varchar (80)	KEY, REQ_ATTR	Task or meeting name.
STORE_ID	LONG, NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Service ID.
CREATION_TIME	DATETIME, NOT_NULL	-	REQ_ATTR, RONLY	Creation time.
STATUS	STATUS_TYPE NOT_NULL	-	REQ_ATTR	Status.
DELETED	LONG NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Deleted status.
LAST_MOD_TIME	DATETIME NOT_NULL	-	REQ_ATTR, RONLY	Last modified time.
“Details” group				
TASK_DESC	TEXT	-	-	Description
START_DATE	DATETIME	-	-	The start date of a task or the date and time of a meeting.
DUE_DATE	DATETIME	-	-	The due date of a task or the end date and time of a meeting.
COMPLETION_DATE	DATETIME	-	-	Task completion date.
PRIORITY	TASK_PRIORITY_TYPE	-	-	Task priority.
TASK_PLAN	TEXT	-	-	Task plan or meeting agenda.
TASK_COMMENT	TEXT	-	-	Task comments or meeting minutes.
ASSIGNED_TO	LONG	-	-	ID of user who owns task or meeting.
ASSIGNED_BY	LONG	-	-	ID of user who last assigned task or meeting.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
AUTHOR	LONG	-	-	The creator of a task or meeting.
LOCATION	STRING	varchar (80)	-	Meeting location.
TASK_STATUS	TASK_STATUS_TYPE	-	-	Task status.
TASK_TYPE	TASK_BASIC_TYPE	-	REQ_ATTR, KEY	Task or meeting.
PERMISSION	ACCESS_TYPE, NOT_NULL	-	REQ_ATTR	The project group in which members can see this task or meeting (OBSOLETE).
PHASE_OID	OID_PHASE	-	REQ_ATTR, KEY	The phase to which the task or meeting belongs (BV_EP_PHASE).
PROJECT_OID	OID_PROJECT	-	REQ_ATTR	The project to which the task or meeting belongs (BV_EP_PROJECT).

BV_EP_TASK_GROUP

Lists the task groups in the project.

Schema file	`ep_task_ext.src`	
Table	BV_EP_TASK_GROUP	This name is required; do not change it.
Parent	BV_EP_TASK	Project's tasks and meetings.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
GROUP_OID	OID_GROUP	-	KEY, REQ_ATTR	Project Group ID

BV_EP_ANNOUNCEMENT

Lists the announcements in the project.

Schema file	`ep_announcement_base.src`	
Table	BV_EP_ANNOUNCEMENT	This name is required; do not change it.
Related to	BV_EP_PROJECT BV_EP_PHASE	Project and project phase tables.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
“Basics” group				
ANNOUNCEMENT_NAME	STRING, NOT_NULL	varchar (80)	KEY, REQ_ATTR	Announcement name.
STORE_ID	LONG, NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Service ID.
CREATION_TIME	DATETIME, NOT_NULL	-	REQ_ATTR, RONLY	Creation time.
STATUS	STATUS_TYPE NOT_NULL	-	REQ_ATTR	Status.
DELETED	LONG NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Deleted status.
LAST_MOD_TIME	DATETIME NOT_NULL	-	REQ_ATTR, RONLY	Last modified time.
“Details” group				
ANNOUNCEMENT_DESC	TEXT	-	-	Description
AUTHOR	LONG	-	-	Creator of the announcement.
PERMISSION	ACCESS_TYPE, NOT_NULL	-	REQ_ATTR	Project group in which members can see this announcement (OBSOLETE).
PHASE_OID	OID_PHASE	-	REQ_ATTR, KEY	Phase to which the announcement belongs (BV_EP_PHASE).
PROJECT_OID	OID_PROJECT	-	REQ_ATTR	Project to which the announcement belongs (BV_EP_PROJECT).

BV_EP_ANN_GROUP

Lists the announcement groups in the project.

Schema file	`ep_announcement_ext.src`	
Table	BV_EP_ANN_GROUP	This name is required; do not change it.
Parent	BV_EP_ANNOUNCEMENT	Project announcements list.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
GROUP_OID	OID_GROUP		KEY, REQ_ATTR	Project Group ID

BV_EP_DFGRP_GROUP

Lists the discussion forum groups in the project.

Schema file	ep_dfgrp_ext.src	
Table	BV_EP_DFGRP_GROUP	This name is required; do not change it.
Parent	BV_DF_GRP	The project's discussion groups.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
GROUP_OID	OID_GROUP		KEY, REQ_ATTR	Project Group ID

BV_EP_GROUP

List the project groups.

Schema files	ep_group_base.src, ep_group_ext.src	
Table	BV_EP_GROUP	This name is required; do not change it.
Child	BV_EP_GROUP_USER	Project group members.
Related to	BV_EP_PROJ_GROUP BV_EP_TASK_GROUP BV_EP_ANN_GROUP BV_EP_DFGRP_GROUP	Project and project task, meeting, announcement, and discussion group tables.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
"Basics" group (ep_group_base.src)				
GROUP_NAME	STRING, NOT_NULL	varchar (80)	KEY, REQ_ATTR	Group Name.
STORE_ID	LONG, NOT_NULL		REQ_ATTR, RONLY, HIDDEN	Interactive Service ID (Read-only).
CREATION_TIME	DATETIME, NOT_NULL		REQ_ATTR, RONLY	Date/Time group was created (Read-only).
STATUS	STATUS_TYPE NOT_NULL		REQ_ATTR	Status.
DELETED	LONG NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Remove From/Keep in Database

Name	Type	Column	Attribute Kind	Friendly name or Semantics
LAST_MOD_TIME	DATETIME NOT_NULL	-	REQ_ATTR, RONLY	Date/Time of Last Modification to Group (Read-only).
“Details” group (ep_group_ext.src)				
GROUP_DESC	TEXT	-	-	Description.

BV_EP_GROUP_USER

List the group members.

Schema file	ep_group_ext.src	
Table	BV_EP_GROUP_USER	This name is required; do not change it.
Parent	BV_EP_GROUP	This name is required; do not change it.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
USER_ID	LONG, NOT_NULL		REQ_ATTR, KEY	User ID

Message attachment tables

The table listed here contains information for handling message attachments.

BV_DF_FILES

Lists the files attached to the message.

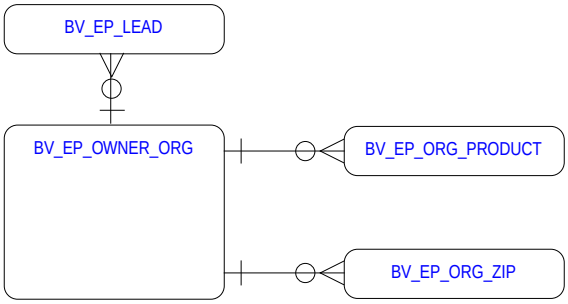
Schema file	ep_dfmsg_ext.src	
Table	BV_DF_FILES	This name is required; do not change it.
Parent	BV_DF_MSG	Discussion group message table.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
FILE_PATH	FILE_PATH, NOT_NULL	varchar (250)	KEY, REQ_ATTR	File path relative to the HTTP document root where the file was uploaded.
DESCRIPTION	TEXT	-	-	File description.
FILE_SIZE	LONG	-	-	File size in bytes.

Lead management tables

The tables listed here contain information for tracking sales leads.



BV_EP_LEAD

Lists lead attributes.

Schema file	`ep_lead_base.src` and `ep_lead_base_ext.src`	
Table	BV_EP_LEAD	This name is required; do not change it.
Related to	BV_EP_OWNER_ORG	Lead owner organization.

The following attributes are required, except for the custom lead attributes.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
“Basics” group (`ep_lead_base.src`)				
LEAD_ID	STRING, NOT_NULL	varchar (80)	KEY, REQ_ATTR	Lead ID.
STORE_ID	LONG, NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Service ID.
CREATION_TIME	DATETIME, NOT_NULL	-	REQ_ATTR, RONLY	Creation time.
STATUS	STATUS_TYPE NOT_NULL	-	REQ_ATTR	Available/Unavailable for Display on Web Site.
DELETED	LONG NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Deleted status.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
LAST_MOD_TIME	DATETIME NOT_NULL	-	REQ_ATTR, RONLY	Last modified time.
“Details” group (ep_lead_base.src)				
DEF_OWNER_ID	LONG	-	-	Default owner ID.
DEF_OWNER_ORG_OID	LONG	-	-	Default owner organization OID for which the default owner is the contact person (BV_EP_OWNER_ORG).
OWNER_ORG_OID	LONG	-	-	Owner organization OID for which the current owner of the lead is the contact person. The current owner ID is in the CP_ASSIGNED_TO attribute.
MANAGER_ID	LONG	-	-	Manager ID of the sales manager for the owner organization of the current owner of the lead.
COMMENT_FOR_OWNER	TEXT	-	-	Comments for owner.
NEXT_SCHED_TIME	DATETIME	-	-	Next scheduled time for alerts.
REMINDER_COUNTER	LONG	-	-	Number of times to remind an owner that this lead is waiting in the owner’s lead in-box.
HRS_TO_NEXT_REMIND	LONG	-	-	Hours until next reminder.
HRS_TO_AUTO_ASSIGN	LONG	-	-	Hours until lead is automatically assigned to the default owner.
“Contact information” group (ep_lead_base.src)				
NAME	STRING, NOT_NULL	varchar (80)	REQ_ATTR	Name of the visitor who generated the lead.
EMAIL	STRING	varchar (80)	-	E-mail address of the visitor.
CONTACT_METHOD	METHOD	int	-	Contact method indicating how the visitor wishes to be contacted, for example, by e-mail, phone, or fax.
“Custom lead attributes” group (ep_lead_spec_ext.src)				
FAX_NUMBER	STRING	varchar (255)	-	Fax number of the visitor who generated the lead.
PHONE_NUMBER	STRING	varchar (255)	-	Phone number of the visitor who generated the lead.
BUSINESS_NAME	STRING	varchar (80)	-	Business name of the visitor who generated the lead.
COUNTRY	STRING	varchar (20)	-	Country of the visitor who generated the lead.
ZIP	STRING	varchar (20)	-	Zip code of the visitor who generated the lead.
PRODUCT	STRING	varchar (80)	-	Product in which the visitor is interested.
QUANTITY	LONG	-	-	Quantity of the product desired by the visitor.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
"Workflow" group (cp_cnt_schema_spec.src) - All of the Publishing Center extended attributes.				
CP_WORKFLOW_STATE	LONG	-	RONLY, HIDDEN	The current workflow state.
CP_ASSIGNED_TO	LONG	-	RONLY, HIDDEN	The ID of the user or access group that currently owns this content.

For more Workflow attributes see the *Publishing Center Developer's Guide, Database Schema Extensions, Workflow extensions.*

When adding attributes to BV_EP_LEAD, to match with attributes in BV_EP_OWNER_ORG, you must also add corresponding attributes to the profile table for those lead attributes.

For each new lead, the following script generates a "guest" user profile. Then the script copies the lead attributes from the leads table to the guest user profile. The script then runs the "Find Owner Org" rule set to match the guest profile to a lead owner organization. See Customizing Closed-loop Process Management for more on adding attributes.

```
$BV1TO1_VAR/msg_scripts/ep_matching_alert.jsp
```

BV_EP_OWNER_ORG

Lists lead owner organizations.

Schema file	ep_owner_org_base.src, ep_owner_org_ext.src	
Table	BV_EP_OWNER_ORG	This name is required; do not change it.
Related to	BV_EP_LEAD	Sales leads table.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
"Basics" group (ep_owner_org_base.src)				
OWNER_ORG_ID	STRING, NOT_NULL	varchar (80)	KEY, REQ_ATTR	Lead owner organization ID.
STORE_ID	LONG, NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Service ID.
CREATION_TIME	DATETIME, NOT_NULL	-	REQ_ATTR, RONLY	Creation time.
STATUS	STATUS_TYPE NOT_NULL	-	REQ_ATTR	Status.
DELETED	LONG NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Deleted status.
LAST_MOD_TIME	DATETIME NOT_NULL	-	REQ_ATTR, RONLY	Last modified time.
"Details" group (ep_owner_org_ext.src)				

Name	Type	Column	Attribute Kind	Friendly name or Semantics
ORG_NAME	STRING	varchar (50)	-	Organization name.
OFFICE	STRING	varchar (50)	-	Office.
ADDRESS1	STRING	varchar (255)	-	Address 1.
ADDRESS2	STRING	varchar (255)	-	Address 2.
CITY	STRING	varchar (50)	-	City.
STATE	STRING	varchar (20)	-	State.
ZIP	STRING	varchar (20)	-	Zip code.
COUNTRY	STRING	varchar (20)	-	Country.
BUS_PHONE	STRING	varchar (255)	-	Business phone.
CONTACT_USER_ID	LONG	-	REQ_ATTR	Contact user ID for the owner organization employee who owns the leads for the owner organization.
MANAGER_ID	LONG	-	-	User ID of the manager of the host enterprise responsible for the owner organization.

BV_EP_ORG_PRODUCT

Lists the products covered by the lead owner organization. This is a multi-value list table. This table is part of the sample schema and is therefore optional.

Schema file	`ep_owner_org_spec_ext.src`	
Table	BV_EP_ORG_PRODUCT	This name is required.
Parent	BV_EP_OWNER_ORG	Lead owner organizations

Name	Type	Column	Attribute Kind	Friendly name or Semantics
PRODUCT	STRING	varchar (80)	KEY, REQ_ATTR	Product

This does not necessarily correspond to an existing product in the database.

BV_EP_ORG_ZIP

Lists the zip codes covered by the lead owner organization. This is a multi-value list table. This table is part of the sample schema and is therefore optional.

Schema file	ep_owner_org_spec_ext.src	
Table	BV_EP_ORG_ZIP	This name is required.
Parent	BV_EP_OWNER_ORG	Lead owner organizations

Name	Type	Column	Attribute Kind	Friendly name or Semantics
ZIP	STRING	varchar (20)	KEY, REQ_ATTR	Zip code

BV_EP_LEAD_HIS

Contains the lead histories. This is a custom InfoExchange Portal table; it is not a content type table.

Table	BV_EP_LEAD_HIS	This name is required; do not change it.
Related to	BV_EP_LEAD	Sales lead table
Related to	BV_EP_OWNER_ORG	Lead owner organizations

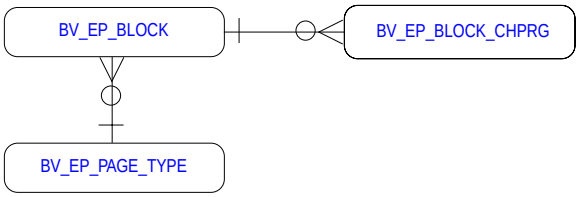
Name	Type	Friendly name or Semantics
LEAD_OID	int not null	Lead content OID.
PREV_OWNER_ID	int default 0	Last owner's USER_ID.
CURR_OWNER_ID	int default 0	Current owner's USER_ID.
PREV_OWNER_ORG_OID	int default 0	Last owner's owner org.
CURR_OWNER_ORG_OID	int default 0	Current owner's owner org.
LEAD_STATE	int default 0	Current workflow state.
LOG_TIME	date default sysdate	Time log entry was recorded.

See "Understanding the lead history table (BV_EP_LEAD_HIS)" on page 181 for more details.

Page type tables

The page type tables list the attributes for page types, blocks, and programs in InfoExchange Portal.

- BV_EP_PAGE_TYPE
- BV_EP_BLOCK
- BV_EP_BLOCK_CHPRG



BV_EP_PAGE_TYPE

Lists page types.

Schema file	`ep_pagetype_base.src`	
Table	BV_EP_PAGE_TYPE	This name is required; do not change it.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
"Basics" group				
PAGETYPE_NAME	STRING, NOT_NULL	varchar (80)	KEY, REQ_ATTR	Page type name.
STORE_ID	LONG, NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Store ID.
CREATION_TIME	DATETIME, NOT_NULL	-	REQ_ATTR, RONLY	Creation time.
STATUS	STATUS_TYPE NOT_NULL	-	REQ_ATTR	On-line status.
DELETED	LONG NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Deleted status.
LAST_MOD_TIME	DATETIME NOT_NULL	-	REQ_ATTR, RONLY	Last modified time.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
“Details” group				
DISPLAY_SCRIPT	STRING	varchar (255)	-	Path to script file relative to the script root for the home page.
NUM_OF_COLUMNS	LONG, NOT_NULL	-	REQ_ATTR	The maximum number of columns to display in the home page.
NUM_OF_ENTRIES	LONG	-	-	The maximum number of entries to display in the required blocks of the page type.

BV_EP_BLOCK

Holds the blocks for a visitor. This is a new content type table.

Schema file	`ep_block_base.src`	
Table	BV_EP_BLOCK	This name is required; do not change it.
Child	BV_EP_BLOCK_CHPRG	Programs within blocks.
Related to	BV_EP_PAGE_TYPE	Page type containing the blocks.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
“Basics” group				
BLOCK_NAME	STRING, NOT NULL	varchar (80)	KEY, REQ_ATTR	Block Name
STORE_ID	LONG, NOT NULL	-	REQ_ATTR, RONLY, HIDDEN,	Service ID
CREATION_TIME	DATETIME, NOT NULL	-	REQ_ATTR, RONLY	Creation Time
STATUS	STATUS_TYPE, NOT NULL	-	REQ_ATTR	On-line status
DELETED	LONG NOT_NULL	-	REQ_ATTR, RONLY, HIDDEN	Deleted status
LAST_MOD_TIME	DATETIME, NOT_NULL	-	REQ_ATTR, RONLY	Last modification time
“Details” group				
BLOCK_TYPE	LONG, NOT_NULL	-	REQ_ATTR	Block type. The values are 0, optional block, 1, default block, and 2, required block.
PAGETYPE_ID	LONG, NOT_NULL	-	KEY, REQ_ATTR	Page type ID (BV_EP_PAGE_TYPE).

BV_EP_BLOCK_CHPRG

Attributes of the programs within a block.

Schema file	`ep_block_ext.src`	
Table	BV_EP_BLOCK_CHPRG	This name is required; do not change it.
Parent	BV_EP_BLOCK	Block content.
Related to	BV_KM_PROGRAM	Programs for a visitor's block.

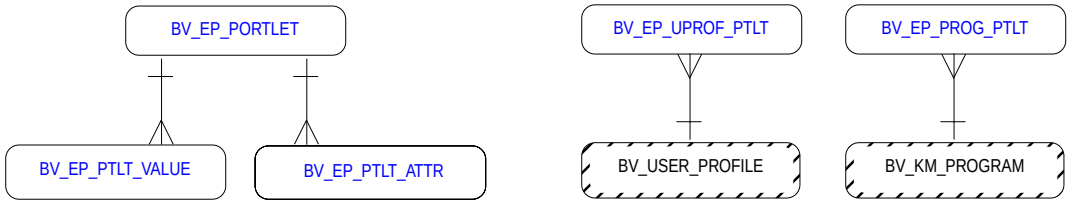
The following attribute is required.

Name	Type	Attribute Kind	Friendly name or Semantics
CHPRG_ID	LONG	KEY, REQ_ATTR	Program OID

Portlet tables

The portlet tables contain the attributes and metadata required for adding portlets to InfoExchange Portal.

- BV_EP_PORTLET
- BV_EP_PTLT_ATTR
- BV_EP_PTLT_VALUE
- BV_EP_UPROF_PTLT
- BV_EP_PROG_PTLT



BV_EP_PORTLET

Lists the metadata and attributes for a portlet.

Schema file	`ep_portlet_base.src`, `ep_portlet_ext.src`	
Table	BV_EP_PORTLET	This name is required; do not change it.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
"Basics" group (`ep_portlet_base.src`)				
PTLT_NAME	STRING, NOT_NULL	varchar(162)	REQ_ATTR	Portlet name (required). This is the name that appears in the Admin Tool.
STORE_ID	LONG, NOT_NULL	REQ_ATTR, RONLY, HIDDEN		Interactive Service ID (Read-only)
CREATION_TIME	DATETIME, NOT_NULL		REQ_ATTR, RONLY	Date/time portlet was created (read-only)
STATUS	STATUS_TYPE, NOT_NULL		REQ_ATTR	On-line (available/unavailable for display on Web site)
DELETED	LONG NOT_NULL		REQ_ATTR, RONLY, HIDDEN	Deleted status (remove from/keep in database)

Name	Type	Column	Attribute Kind	Friendly name or Semantics
LAST_MOD_TIME	DATETIME, NOT_NULL		REQ_ATTR, RONLY	Date/time of last modification to portlet (read-only)
“Details” group (ep_portlet_ext.src)				
PTLT_DESC	STRING	varchar(255)		Portlet description.
PTLT_VERSION	STRING	varchar(80)		Portlet version (optional).
PTLT_BLOCK_SCPT	STRING	varchar(128)		Portlet block script path (required). The path, relative to the script-root, to the script that displays the portlet output to a home page.
PTLT_PAGE_SCPT	STRING	varchar(128)		Portlet page script path. The path, relative to the script-root, to the script that displays the portlet output when browsing the channel/program hierarchy.
PTLT_VIST_SCPT	STRING	varchar(128)		Portlet visitor script path. The path, relative to the script-root, to the script that lets the visitor specify attributes. This script must be the same type of script as specified by PTLT_TYPE.‘
PTLT_REG_FILE	STRING	varchar(128)		Portlet registration file.
PTLT_TYPE	PTLT_TYPE_TYPE	int		Portlet language type (required); must be either JavaScript or JSP.
PTLT_CACHEABLE	BOOLEAN_ENUM, NOT_NULL			Is This Portlet Cacheable (optional). A flag set in the Admin Tool that determines whether the system caches the output of this portlet.
PTLT_IEP_VER	STRING	varchar(20)		InfoExchange Portal version; the version of InfoExchange Portal in which this portlet was created.

BV_EP_PTLT_ATTR

Lists portlet attributes.

Schema file	ep_portlet_ext.src	
Table	BV_EP_PTLT_ATTR	This name is required; do not change it.
Parent	BV_EP_PORTLET	Portlet attributes and metadata.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
ATTR_ID	STRING	varchar(10)	KEY, REQ_ATTR	Attribute ID (required). This value is unique within this portlet but does not have to be unique across portlets. Changing this ID (by reregistering the portlet) is considered a delete-and-add operation.
ATTR_FRNDLY_NAME	STRING	varchar(80)		Attribute Friendly Name. This is the label that appears in the Admin Tool; it can be changed without deleting the attribute.
ATTR_CTRL_TYPE	PTLT_CTRL_TYPE	int		Portlet Attribute Control Type. This determines the HTML control to show in the Admin Tool for this attribute (text, text area, select box, radio button, or check box).
ATTR_SCOPE	PTLT_SCOPE_TYPE	int		Portlet Attribute Scope. Must be one of the following values: 0 = Sitewide 1 = Program-specific 2 = Visitor-specific
ATTR_INITIAL	STRING	varchar(128)		Portlet Attribute Initial Value. For site-wide parameters, this is the actual value for the parameter. For program-specific and visitor-specific parameters, this is the default value and can be overridden by the values specified in the visitor or program list tables.
ATTR_REQUIRED	BOOLEAN_ENUM, NOT_NULL			Is The Value Needed For This Attribute. Valid values are: 0 = Optional 1 = Required Note that the database does not enforce required attributes.

BV_EP_PTLT_VALUE

Lists values of portlet dynamic attribute control types.

Schema file	`ep_portlet_ext.src`	
Table	BV_EP_PTLT_VALUE	This name is required; do not change it.
Parent	BV_EP_PORTLET	Portlet attributes and metadata.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
V_ATTR_ID	STRING	varchar(10)	KEY, REQ_ATTR	Attribute ID (required). This value is unique within this portlet but does not have to be unique across portlets. Changing this ID (by reregistering the portlet) is considered a delete-and-add operation.
V_CTRL_VALUE	STRING	varchar(128)	KEY, REQ_ATTR	Control Type Value. When the HTML allows multiple values for a variable, this determines the value to be shown.

BV_EP_PROG_PTLT

Lists the portlet attributes for a program.

Schema file	`ep_program_ext.src`	
Table	BV_EP_PROG_PTLT	This name is required; do not change it.
Parent	BV_KM_PROGRAM	Portlet attributes and metadata.

The following attributes are required.

Name	Type	Column	Attribute Kind	Friendly name or Semantics
PTLT_ID	PTLT_OID_TYPE, NOT_NULL		KEY, REQ_ATTR	Portlet Object ID.
ATTR_ID	STRING	varchar(10)	KEY, REQ_ATTR	Portlet Attribute ID. The PTLT_ID and ATTR_ID are a composite key. The ATTR_ID must be unique within a portlet.
ATTR_VALUE	STRING	varchar(128)		Portlet Attribute Value.

B Getting Technical Support

Technical Support services are provided on an annual basis to BroadVision customers. A standard 90-day warranty is also provided with all software.

If you experience problems in using any of BroadVision's software products, contact BroadVision's World-wide Customer Support Organization for assistance. Registered customers who have a login name and password can report problems via BroadVision's Web site, `www.broadvision.com`. On the Web site you can access support-related technical information, report problems, and track report status and responses on the Problem Reports pages. To request a login, please contact your BroadVision Account Representative.

The Web site is the preferred method of reporting problems; however, if necessary you can report problems via e-mail to `bvhelp@broadvision.com`. Please be sure to include the case ID when communicating via e-mail.

Information on how to contact Customer Support offices by telephone can be found on the BroadVision Web site.

Support for Third-Party Software Products

To allow for complete testing, BroadVision certifies BroadVision One-To-One products against the versions of third-party products that are released and available sufficiently in advance of the software release date. This often means that third-party vendors release new versions of their products prior to the next release of the BroadVision software. While BroadVision would prefer customers use the tested and certified software versions, we also understand customers will occasionally want or need to use these new versions of third-party products. As long as the vendor guarantees forward compatibility, One-To-One products work on these new versions.

BroadVision usually tests and certifies these newer versions of third-party products in the next product release. This can be a good indicator the newer versions will work with the previous release. In exceptional cases BroadVision may determine the newer version of a third-party product cannot be used because it fails in some way during the testing cycle. In this case we will continue to certify the older version.

BroadVision will support customers who use newer versions of third-party products by working with the customer to resolve compatibility problems with the third-party vendor. BroadVision will also consider, at our option, developing and releasing minor fixes for our products to resolve problems with new versions of third-party products.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X