

# 1 Project Page

This chapter describes how a developer can customize the APIs and template.jsp files for creating and using a Project Page.

An employee, such as a Project Manager, can create a Project Page to share information about a collaboration, or project, with other employees, customers, or partners. Shared information about a project can include tasks, meetings, and announcements. A Project 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.

Typically, a project contains these data types:

- Ordered list of tasks which have an associated title, description, due date, and status (such as complete)
- Time ordered list of meetings, where each has a title, description, meeting date, and meeting minutes section
- Time ordered list of announcements, sorted by creation date, which have an associated title and description
- Contact list with links to people who are contacts for a project, where the list is a subset of all the participants of a collaboration

---

# Project Component APIs

The Developer can use the Project component APIs described in `epprojectmanager.jsi` for customizing the Project Page.

---

## EPPROJECT MANAGER\_JSI

The `epprojectmanager.jsi` API enables the developer to customize the Project Page:

- projects
- phases
- participants
- tasks
- meetings
- announcements

The following script implements the `epprojectmanager.jsi` API:

```
#ifndef _EPPROJECTMANAGER_JSI
#define _EPPROJECTMANAGER_JSI

implementation
{
    include <jsrt/properties_i.hh>
    include <jsrt/valuelist_i.hh>
    include "epprojectmanager_i.hh"
}

interface BVI_EPPProjectManager
{
    implementation { implementation_type = BVC_EPPProjectManager; }
```

The JSI constructors are:

```
creator();
creator(BVI_EPPProjectManager ref);
```

This part of the script returns a copy of the `BVI_EPPProjectManager`, or returns null if an error occurs:

```
BVI_EPPProjectManager clone();
```

## Project Methods

The Project methods enable the Developer to add, update, delete, or get a project.

1. To add a project, use **long addProject**. **BVI\_EPPProject** object returns the project ID for the new project, where **<project>** is the new **BVI\_EPPProject** object created or added.

```
long addProject(string serviceId,  
               long userId,  
               BVI_EPPProject project );
```

2. To update a project, use **void updateProject**. **<project>** is the **BVI\_EPPProject** object changed or updated.

```
void updateProject(string serviceId,  
                  long userId,  
                  BVI_EPPProject project );
```

3. To delete a project, including tasks, meetings, and announcements, use **void deleteProject**. **<projectId>** is the project's ID or name deleted.

```
void deleteProject(string serviceId,  
                  long userId,  
                  string projectId );
```

4. To delete an entire project, use **void deleteEntireProject**. When deleting an entire project, all the phases, participants, related products, discussion groups and attachments are deleted. **<projectId>** is the project's ID or name deleted.

```
void deleteEntireProject(string serviceId,  
                        long userId,  
                        string projectId );
```

5. To get a project by project identification or project name, use **BVI\_EPPProject getProject**, which returns a **BVI\_EPPProject** object. **<projectId>** is the project's ID or name.

```
BVI_EPPProject getProject(string serviceId, string projectId);
```

6. To locate all user-related projects, use **BVI\_ValueList findUserProject** to return the list of projects for **<userId>**. A **BVI\_EPPProject** object is set in each **BVI\_Value** returned in the value list. Use **BVI\_Value::objectValue** to reference the **BVI\_EPPProject** object.

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

## Phase Methods

The Phase methods enable the developer to add, update, delete, or get phases.

1. To add a phase, use **long addPhase**. This script returns the New phase Id for the new phase, where **<phase>** is the **BVI\_EPPPhase** object created or added.

```
long addPhase(string serviceId,  
long userId,  
BVI_EPPPhase phase);
```

2. To update a phase, use **void updatePhase** to update a phase, where **<phase>** is the **BVI\_EPPPhase** object changed or updated.

```
void updatePhase(string serviceId,  
long userId,  
BVI_EPPPhase phase);
```

3. To delete a phase, use **void deletePhase**. **<phaseId>** is the phase's ID deleted.

```
void deletePhase(string serviceId,  
long userId,  
string phaseId );
```

4. To delete an entire phase, as well as the tasks, meetings, and announcements in the phase, use **void deleteEntirePhase**. **<phaseId>** is the phase's ID deleted

```
void deleteEntirePhase(string serviceId,  
long userId,  
string phaseId );
```

5. To get a phase object by **phase\_id**, use **BVI\_EPPPhase getPhase**, which returns a **BVI\_EPPPhase** object. **<phaseId>** is the phase's ID.

```
BVI_EPPPhase getPhase(string serviceId, string phaseId);
```

6. To get all project-related phases, use **BVI\_ValueList getPhaseInProject**, which returns the list of phases for **<projectId>**. A **BVI\_EPPPhase** object is set in each **BVI\_Value** returned in the value list. Use **BVI\_Value::objectValue** to reference the **BVI\_EPPPhase** object.

```
BVI_ValueList getPhaseInProject(string serviceId,  
string projectId);
```

## Participant Methods

These Participant methods enable the Developer to add, update, delete, or get participants.

1. To add a participant, use **long addParticipant**, which returns the status.

```
long addParticipant(string serviceId,
long userId,
string projectId,
boolean contact,
boolean owner,
string contactInfor);
```

- **<contact>**—true if the participant is a contact person of the project
- **<owner>**—true if the participant is a owner of the project
- **<contactInfor>**—title of the participant

2. To update a participant, use **long updateParticipant**, which returns the status.

```
long updateParticipant(string serviceId,
long userId,
string projectId,
boolean contact,
boolean owner,
string contactInfor);
```

- **<contact>**—true if the participant is a contact person of the project
- **<owner>**—true if the participant is a owner of the project
- **<contactInfor>**—title of the participant

3. To delete a participant, use **long deleteParticipant**, which returns the status.

```
long deleteParticipant(string serviceId,
long userId,
string projectId);
```

4. To get all project-related participants, use **BVI\_ContentList getParticipantInProject**, which returns the list of participants including userId, owner, contact and contact Information for **<projectId>**. The return list is a **BVI\_ContentList**.

```
BVI_ContentList getParticipantInProject(string serviceId,
string projectId);
```

5. To get all project-related contacts, use **BVI\_ContentList getContactInProject**, which returns the list of contact persons including userId, and contact information for **<projectId>**. The return list is a **BVI\_ContentList**.

```
BVI_ContentList getContactInProject(string serviceId,
string projectId);
```

## Task Methods

These Task methods enable the Developer to find tasks in a phase and in a project.

- To find all tasks in a phase, use **BVI\_ContentList getTaskInPhase**, which returns the list of tasks for **<userId>** and **<phaseId>**. The return list is a **BVI\_ContentList**.

```
BVI_ContentList getTaskInPhase(string serviceId,  
long userId,  
string phaseId);
```

- To find all tasks in a project, use **BVI\_ContentList getTaskInProject**, which returns the list of tasks for **<userId>** and **<projectId>**. The return list is a **BVI\_ContentList**.

```
BVI_ContentList getTaskInProject(string serviceId,  
long userId,  
string projectId);
```

## Meeting Methods

These Meeting methods enable the Developer to find meetings in a phase and in a project.

- To find all meetings in a phase, use **BVI\_ContentList getMeetingInPhase**, which returns the list of meetings for **<userId>** and **<phaseId>**. The return list is a **BVI\_ContentList**.

```
BVI_ContentList getMeetingInPhase(string serviceId,  
long userId,  
string phaseId);
```

- To find all meetings in a project, use **BVI\_ContentList getMeetingInProject**, which returns the list of meetings for **<userId>** and **<projectId>**. The return list is a **BVI\_ContentList**.

```
BVI_ContentList getMeetingInProject(string serviceId,  
long userId,  
string projectId);
```

## Announcement Methods

These Announcement methods enable the Developer to locate announcements in a phase and in a project.

- To locate all announcements in a phase, use **BVI\_ContentList getAnnouncementInPhase**, which returns the list of announcements for **<userId>** and **<phaseId>**. The return list is a **BVI\_ContentList**.

```
BVI_ContentList getAnnouncementInPhase(string serviceId,  
long userId,  
string phaseId);
```

- To locate all announcements in a project, use **BVI\_ContentList getAnnouncementInProject**, which returns the list of announcements for **<userId>** and **<projectId>**. The return list is a **BVI\_ContentList**.

```
BVI_ContentList getAnnouncementInProject(string serviceId,  
long userId,  
string projectId);
```

## Miscellaneous Methods

- To check if the user is the project's owner, use **boolean isOwnerofProject**, which returns a true status if the user is the project's owner.

```
boolean isOwnerofProject(string serviceId,
long userId,
string projectId);
```

- To delete all content-related attachments for the item, use **long deleteAllFiles**, which returns the status **contentOid** as the content OID.

```
long deleteAllFiles(long contentOid);

long deleteAllFilesInPath( string relativeDir );

}

#endif
```

---

## EPPROJECT\_JSI

The **epproject\_jsi** API enables the Developer to customize projects within the Project Page.

The following script implements the **epproject\_jsi** API:

```
#ifndef _EPPROJECT_JSI
#define _EPPROJECT_JSI

implementation
{
    include <jsrt/datetime_i.hh>
    include <jsrt/properties_i.hh>
    include <jsrt/valuelist_i.hh>
    include "epproject_i.hh"
}
{

    implementation { dynamicProperties = false; }
```

The Developer can change the following attributes for strings in **interface BVI\_EPPProject** to modify the project interfaces:

Attribute String	Description
<b>long projectId</b>	Project ID (read only).
<b>projectName</b>	Project name.
<b>projectDescription</b>	Project description.
<b>projectGoal</b>	Project goal.
<b>projectPagepath</b>	Page path.
<b>long projectCurrentphase</b>	Project's current phase.
<b>projectIconpath</b>	Icon path.

Attribute String	Description
<code>long status</code>	Content status: 0—Offline 1— On-line
<code>strcol1</code>	Developer can modify the first column.
<code>strcol2</code>	Developer can modify the second column.
<code>strcol3</code>	Developer can modify the third column.

The JSI constructors are:

```
creator();
creator(BVI_EPPProject ref);
```

This part of the script returns a copy of the `BVI_EPPProject`, or returns null if an error occurs:

```
BVI_EPPProject clone();
}
```

## EPPHASE\_JSI

The EPPROJECT\_JSI API enables the Developer to customize phases within the Project Page.

The following script implements the EPPROJECT\_JSI API:

```
#ifndef _EPPHASE_JSI
#define _EPPHASE_JSI

implementation
{
    include <jsrt/datetime_i.hh>
    include <jsrt/properties_i.hh>
    include <jsrt/valuelist_i.hh>
    include "epphase_i.hh"
}
{
    implementation { dynamicProperties = false; }
```

1. The Developer can change the following attributes for strings in `interface BVI_EPPPhase` to modify the project interfaces:

Attribute String	Description
<code>long phaseId</code>	Phase ID (read only).
<code>phaseName</code>	Phase name.
<code>phaseDescription</code>	Phase description.
<code>phaseGoal</code>	Phase goal.
<code>long projectID</code>	Project ID of the phase.

Attribute String	Description
<code>strcol1</code>	Developer can modify the first column.
<code>strcol2</code>	Developer can modify the second column.
<code>strcol3</code>	Developer can modify the third column.

The JSI constructors are:

```
creator();  
creator(BVI_EPPHase ref);
```

This part of the script returns a copy of the `BVI_EPPHase`, or returns null if an error occurs:

```
    BVI_EPPHase clone();  
}
```