Jive Anywhere – Cartridges SDK 2.0

Overview

Jive Anywhere allows extending its behavior by creating and uploading cartridges via the Admin console. For instance a cartridge can change the search logic, change UI elements and provide a method for generating previews for created discussions. Each cartridge is activated for a list of predefined URL patterns.

You can develop a cartridge for your company needs, such as extending a web app or download a cartridge created by others for customizing the behavior of a well-known website. Cartridges are developed using industry standards and easy to learn technologies including JavaScript, HTML and CSS.

Minimum requirements

Client requirements:

- Jive Anywhere 2.0
- Internet Explorer 8, Chrome 22, Firefox 16 or Safari 5.1

Server requirements:

- Extended APIs 2.1
- Jive 5.x

When Jive Anywhere is connected to a server running Extended APIs earlier than v2.1, it will run in compatibility mode and cartridges will be disabled.

When using multiple communities, cartridges are only available for the default community. The default community can be set using the settings page of Jive Anywhere.

Common use cases for the SDK

The SDK can extend Jive Anywhere and also enrich third party websites with Jive content. Here is a list of the most common use cases can be achieved with the SDK:

- Create a custom dynamic preview which is added as a quote when posting a discussion. By default OpenGraph data is extracted from the web page but you can easily replace this logic with custom HTML template bound to any part of the DOM. For instance the included LinkedIn cartridge adds a preview of the visited profile into the discussion.
- Enrich third party websites with content from Jive. For instance you can write code to show a user popup with details from Jive when hovering a user hyperlink in LinkedIn.

- Define a discussion list search logic for a website. For instance the included LinkedIn cartridge extracts the visited profile name from LinkedIn and performs a search for that name in Jive. The search results are then displayed in the sidebar instead of just searching for discussions contain the URL string.
- Bring content from other servers into the search results. For instance you can query other intranet web servers in the organization using standard HTTP requests and show the results in the Jive Anywhere sidebar.
- Change the logic used for connecting a web page to a group, space or project. By default the connection is made for just the visited URL but in some cases you want to connect the whole website or all pages related to the same logical subject. After the logic was defined, whenever a user is connecting a group, the same group will also appear as connected to all other pages defined by this logic.
- Change some UI elements in the Jive Anywhere sidebar when visiting specific web pages. For instance you can change the title of the "Community" and "Places" tabs, set the default new discussion title to be used, and append a container with additional HTML content to the search results.

Developer Guide

Components

A cartridge is composed by the following components:

- module.js is the cartridge's main script file that runs in the background context of Jive Anywhere. It can implement any or none of the pre-defined APIs to set runtime hooks. Cross-origin HTTP requests can be initiated here.
- pagescript.js (optional) is a script file that is injected into the page when the cartridge is active. It can contain public helper functions for reading and manipulating the DOM. Messages can be sent and received between module.js and this script.
- icon.png (optional) is a 16x16 pixels image file that is displayed on the notification button when the cartridge is active.
- Other content files (optional) such as images, HTML templates, CSS files and so on. These files can be read by using **ModuleContext.getResourceFile()** and referred to by using **ModuleContext.getResourceUrl()** which returns a fully qualified URL that can be embedded into the HTML.
- Some metadata, such as the cartridge display name and a list of activation URL patterns. These are provided in the Admin console.

Creating your first Jive Anywhere cartridge

To create a basic LinkedIn cartridge, create an empty folder in your local file system with the following files:

module.js

```
// Note: ModuleContext and $ are provided by the module manager
// The init function runs only if the cartridge has been activated
this.init = function () {
   // Compile markup as named templates to be used by onGetPreviewData later
```

```
   var html = ModuleContext.getResourceFile("LinkedInPageDataTemplate.html");
   $.templates({
      linkedInPageDataTemplate: html
   });
};

// Invoked on URL (community) search
this.onGetUrlSearchResults = function (search, callback) {
   // Extracts the username from the page title
   var userName = extractUserName();

  if (userName != null) {
      // Performs an exact search based on the extracted username and pass the results
      // to the callback
      var query = "\"" + userName + "\"";
      ModuleContext.clientFacade.search(query, search.offset, 20, search.sortBy,
search.isAscending, callback);
   }
};

// Display the username as the title of the Community tab
this.onGetModuleUI = function (callback) {
   var moduleUiInfo = { defaultTabId: 0, tabs: [{ title: extractUserName() }, {}] };
   callback(moduleUiInfo);
};

// Generates preview data when submitting a new discussion instead of the default OpenGraph
this.onGetPreviewData = function (openGraphMetadata, callback) {
   // Extracting page data by invoking a page helper function on pagescript.js
   ModuleContext.runPageScript("getPreviewData", null, function (pageData) {
      // Use the LinkedInPageDataTemplate.html template to generate html markup
      var html = $.render.linkedInPageDataTemplate(pageData);
      // Pass the html back to the module manager
      callback(html);
   });
};

// Helper function to extract the username from the page title
var extractUserName = function () {
    var title = ModuleContext.pageInfo.title;
    return $.trim(title.substring(0, title.indexOf("| LinkedIn")));
};
```

pagescript.js

```
// PageModuleContext and $ are provided by the module manager
// Page helper function called by onGetPreviewData to extract fields out of the page
this.getPreviewData = function (params, sendResponse) {
    var profile = new Object();

    profile.fields = [];
    profile.url = document.location.href;
    profile.name = clean($(".full-name").text());
    profile.locality = clean($(".locality:eq(0)").text());
    profile.industry = clean($(".industry:eq(0)").text());

    if ($(".profile-overview").length > 0) {
        // new LinkedIn site
        profile.image = $(".profile-picture img").first().attr("src");
        profile.title = clean($("p.title:eq(0)").text());

        $(".profile-overview table").find("th").each(function (index) {
            var title = clean($(this).text());
            if (title != "Websites") {
                var data = $(this).next("td").find("li");
                var values = [];
                data.each(function () {
                    var value = clean($(this).text());
                    if (value.substring(0, 1) == ",") {
                        value = value.substring(1);
                    }
                    values.push(value);
                });
                if (data.length > 0) {
                    profile.fields.push({ name: clean(title), values: values });
                }
            }
        });
    }

    if (!profile.image) {
        // use default profile image if image not defined
        profile.image =
"http://static02.linkedin.com/scds/common/u/img/icon/icon_no_photo_40x40.png";
    }

    sendResponse(profile);
};


// Remote whitespaces from the given text
var clean = function (txt) {
    var tmp = $.trim(txt.replace(/\n/g, "").replace(/\t/g, "")).split(" ");
    var tmp2 = [];
    for (var i = 0; i < tmp.length; i++) {
        if (tmp[i] != "") {
            tmp2.push(tmp[i]);
        }
    }

    return tmp2.join(" ");
};
```

LinkedInPageDataTemplate.html

```
<div style="margin-top: 5px; background-color: #fbfbfb; border: 1px solid #D6D6D6;">
    <table style="border: 0px solid #CCC;">
        <tr>
            <td>
                <a href="{{>url}}" target="_blank">
                    <img src="{{:image}}" alt="Profile image" style="border: 1px solid #D1D1D1;" />
                </a>
            </td>
            <td>
                <b><a href="{{>url}}" target="_blank">{{>name}}</a></b><br />
                  {{>title}}<br />
                  {{>locality}} | {{>industry}}
            </td>
        </tr>
      {{for fields}}
        <tr>
            <td>{{>name}}</td>
            <td>
                {{for values}}
                   <div>{{>#data}}</div>
                {{/for}}
            </td>
        </tr>
      {{/for}}
    </table>
</div>
```

Note: Any text editor can be used such as Notepad or Visual Studio for syntax highlighting.

Download the image file located at https://developer.linkedin.com/sites/default/files/LinkedIn_Logo16px.png and save it as **icon.png** into the same folder.

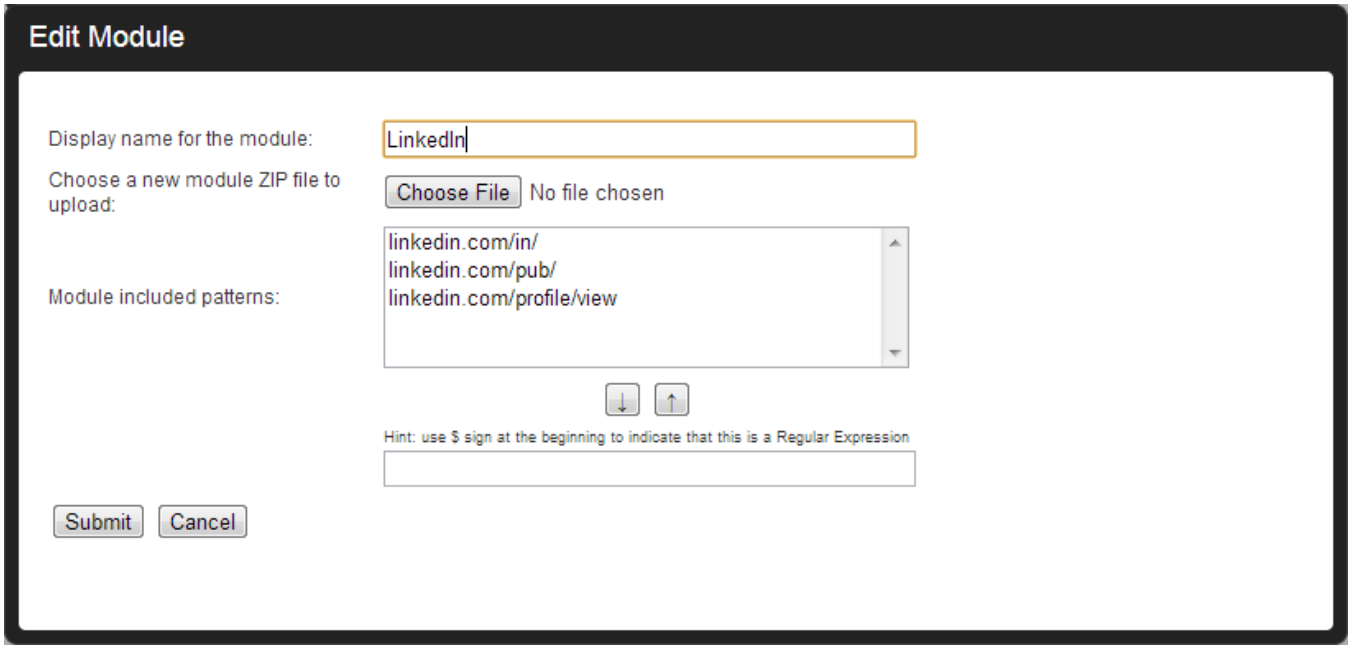
Pack the content of the folder into a zip file named **LinkedInModule.zip** (any name is okay).

Head to the Admin console of your Jive SBS (http://*your-jive-address*/admin), select **System -> Settings -> Jive Anywhere** at the menus and click the **Modules** tab. You must have an account with administrator privileges to enter this page.

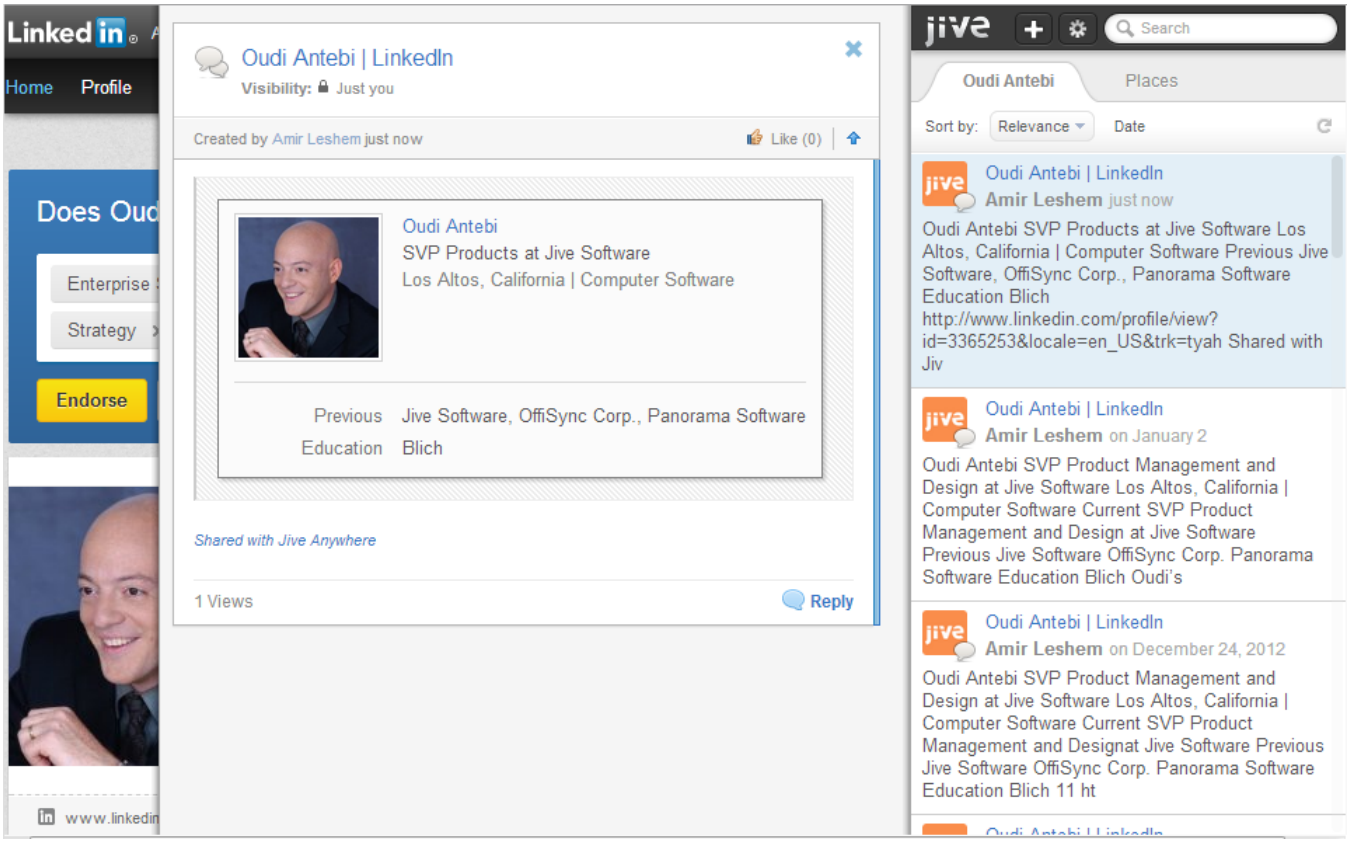
Click **Upload new module**. In the **display name** field type LinkedIn. Click “Choose file” and select the created LinkedInModule.zip file.

In the **included patterns** field type *linkedin.com/profile/view* and click the **up** button to move it into the patterns list. Repeat this step with *linkedin.com/pub/* and *linkedin.com/in/*. The created cartridge will be activated only for URLs matching these patterns.

When finished click **Submit** and make sure you receive no errors.



Restart your browser and head to a profile page on LinkedIn. You should now see the LinkedIn icon appears on the Jive Anywhere sidebar. Open the sidebar, the profile name should appear as the title of the Community tab. Create a new discussion, choose any participant (or location) and click **Post**. Click the created discussion and you should see a preview of the LinkedIn profile embedded into its body.



API Reference

Script files

module.js

This is the cartridge's main script file that runs in the background context of Jive Anywhere. It can implement any or none of the following public callbacks to set runtime hooks. Unimplemented callbacks will provide the default behavior of Jive Anywhere.

The script can use the provided **ModuleContext** helper API. In addition, **jQuery** and **JsRender** are available (using the **$** keyword) out of the box. The script can freely use cross-origin Ajax calls using the standard XmlHttp object or jQuery. Jive specific services should be called using the provided **ModuleContext.clientFacade** which automatically handles authentication and errors.

- init ()
  Invoked after the cartridge has been activated and pagescript.js has been injected into the page.
- onGetModuleUI (callback) **new in 2.0**
  Invoked after the sidebar is opened. Use the callback to returns ModuleUiInfo object, which contains the default tab id, default new discussion title, additional HTML notification to display for each tab and the title of each tab.

- onMessage (command, data, sendResponse)
  Invoked when **PageModuleContext.sendMessage** is called by pagescript.js.
- onGetUrlSearchResults (searchData, callback)
  Invoked when a URL (community) search is in progress. It can be overridden to replace the default search logic and display different search results.
- onGetContainerSearchResults (searchData, callback)
  Invoked when a connected place (group/space/project) search is in progress. It can be overridden to replace the default search logic and display different search results.
- onGetNormalizedSearchUrl (url)
  Invoked after the sidebar is opened. Returns the normalized URL of a visited page based on the original one. This function is useful for cleaning the actual URL being searched and attached to a new discussion or to generate a virtual URL for single-address sites.
- onGetRelatedContainerTag ()
  Invoked when connecting, disconnecting or getting the connected place(s) for the page. Returns the key tag for connecting a place to a URL. Identical tag keys for different URLs will handle the same connected place(s).
- onGetPreviewData (openGraphMetadata, callback)
  Invoked when a new discussion is created. Returns the HTML preview for the discussion. If not supplied a preview based on OpenGraph will be used. You can use the openGraphMetadata dictionary object for creating your own html or invoke a page context helper to extract additional data.
- onCreatePublicDiscussion (discussion, callback) **new in 2.0**
  Invoked when a new public discussion should be created
- onCreatePrivateDiscussion (discussion, callback) **new in 2.0**
  Invoked when a new private discussion should be created

pagescript.js

This script file is injected into the page when the module is active, prior to calling the cartridge's **init()**. It can contain public helper functions for reading and manipulating the DOM. These functions are defined by the *(object params, func sendResponse)* signature and can be invoked by module.js using **ModuleContext.runPageScript()**.

The script can use the provided **PageModuleContext** helper API. In addition, **jQuery** is available (using the **$** keyword) out of the box.

Container types and Helper classes

ModuleContext

Used by module.js to provide a set of helper APIs.

- *PageInfo* pageInfo
  Returns information about the current visited web page.
- *ClientFacade* clientFacade
  Returns a communication proxy to the logged-in Jive server.

- runPageScript (*string* funcName, *object* params, callback)
  Runs the given function on the page context of previously injected pagescript.js, passing the params object. When sendResponse(result) is called by the targeted function, callback is invoked with the result as a parameter.
- getResourceFile (url)
  Gets the textual content of a file from the cartridge package. The provided URL should be relative to the cartridge's root folder.
- getResourceUrl (url)
  Gets a fully qualified URL of a resource file from the cartridge package. The provided URL should be relative to the cartridge's root folder. The returned URL can then be used as a source for image elements or other embedded content.
- getConnectionSettings()
  Gets current connection settings including serverUrl, isConnected and embeddedAuthToken.
- newTab (url, isBackgroundTab) **new in 2.0**
  Opens a new tab with the specified URL. If isBackgroundTab is true, the tab will not be activated automatically.

PageModuleContext

Used by pagescript.js to provide a set of helper APIs.

- sendMessage (*string* command, *object* data, callback)
  Sends a command name and data object to the cartridge's onMessage function. When sendResponse(result) is called by the onMessage function, callback is invoked with the result as a parameter.
- createDiscussionFrame (frameId, typeId, itemId) **new in 2.0**
  Create a new IFRAME with the loaded Jive item specified by the typeId and itemId. The IFRAME can then be appended into the DOM. Its height is managed automatically by Jive Anywhere so you'll never see scrollbars inside the frame. When the close button is clicked, the public function `discussionClosed()` is invoked on pagescript.js so you can hide or remove the IFRAME from the DOM.
- showSidebar(shouldOpenNewDiscussion) **new in 2.0**
  This function allows you to open the sidebar when necessary and optionally open the new discussion dialogue (for instance when a custom button is clicked)

JCommon new in 2.0

A single-ton helper class which provides many common functions. It is available for both module.js and pagescript.js contexts.

- trimHtmlWhitespaces(html)
- htmlEncode(value)
- htmlDecode(value)
- getNodeSource(htmlElement)
- getNodeInnerSource(htmlElement)
- shortenString(str, maxLength)
- getCookie(name)

- getQueryStringParameterByName(name)
- getNonRelativeUrl(url)
- asPrettyDate(date)
- addCssFile(url)
- isIE
- isOldIE
- isQuirksMode

Jive-specific functions

- convertServerDate(serverDate)
- getJiveObjectTypeById(id)

PageInfo

Used by module.js to provide details about the current visited web page.

- url
- title
- domain
- bool isFrameSet

ModuleUiInfo

Used by module.js to get an object that defines custom UI changes to the sidebar. All properties are optional, if a value is not supplied the default one will be used.

- defaultTabId
  (optional) The default tab to display when opening the sidebar in activated page (0 = discussions, 1 = related).
- defaultTitle
  (optional) The default title for the new discussion form
- hideNewDiscussion **new in 2.0**
  (optional) Set to true if you wish to hide the new discussion button
- hideSnapshots **new in 2.0**
  (optional) Set to true if you wish to hide the "add a new snapshot" button
- tabs[] this array must contain 2 objects, the first one is for the "Discussions" tab and the second for the "Related" tab
  - html
    (optional) HTML of custom container displayed at the top of the search results when the module is active.
  - title
    (optional) The search title to display replacing the generic this page term, "Viewing discussion about **this page**".

ClientFacade

This class can be used to access Jive services REST APIs. Authentication and errors are handled automatically.

- searchUrl (url, offset, limit, sortBy, isAscending, callback)
  Use to search for items based on URLs. Call on onGetUrlSearchResults and onGetGroupSearchResults to create custom queries and manipulating results before sending them to the callback function of the event.
- search (query, offset, limit, sortBy, isAscending, callback)
  Use to search for items based on text content. Call on onGetUrlSearchResults and onGetGroupSearchResults to create custom queries and manipulating results before sending them to the callback function of the event.
- getContentInContainer (containerTypeId, containerId, offset, limit, filterType, callback)
  Use to retrieve content in group, space or project.
- getConnectedUser (callback)
  Returns the current connected user.
- getUserById (id, callback)
  Returns a user matches the id.
- searchMentions (query, callback)
  Returns results for @mention query.
- searchSpotlight (query, offset, limit, callback)
  Returns lists of places, people and content for the given query.
- searchUsers (query, limit, callback)
  Returns list of people for the given query.
- searchContainers (query, limit, callback)
  Returns list of places for the given query.
- createCommentForItem (itemTypeId, itemId, htmlContent, callback)
  Creates a new comment or reply to the given item.
- createPublicDiscussion (subject, htmlContent, isQuestion, containerTypeId, containerId, callback)
  Creates a new public discussion at the specified container (place).
- createPrivateDiscussion (subject, htmlContent, isQuestion, userIds, callback)
  Creates a new private discussion between you and the given users.
- getConnectedContainers (tag, callback)
  Returns a list of connected places to a tag.
- attachContainerToTag (containerTypeId, containerId, tag, callback)
  Attaches a place into a new or existing tag.
- detachContainerFromTag (containerTypeId, containerId, tag, callback)
  Detaches a place from a tag.
- createNewConnectedGroup (name, tag, description, groupType, callback)
  Creates a new group attached to the given tag.
- sendGroupInvites (groupId, groupName, userEmails, callback)
  Sends group invites to the given emails.
- getDefaultDiscussionFolders (callback)
  Returns the recent containers used to create a discussion.
- getDiscussion (typeId, itemId, increaseViewCount, callback)
  Returns a discussion and its comments.

- like (typeId, itemId, callback)
  Sets the given item as liked.
- unlike (typeId, itemId, callback)
  Sets the given item as unliked.
- markCorrect (messageId, boolToggleState, callback)
  Marks the given item as correct or incorrect according to the toggle state.
- markHelpful (messageId, boolToggleState, callback)
  Marks the given item as helpful or un-helpful according to the toggle state.
- restHandler **new in 2.0**
  Call restHandler.sendWebRequest (requestType, actionUrl, *object* data, callback, *bool* noCredentials?, applicationName?, bool isJson?) to send an Ajax request using the underlying communication layer. requestType can be either "GET", "POST", "HEAD" or "DELETE". actionUrl is a URL for the service related to the server address. Error handling and authentication is managed by Jive Anywhere. If you wish to send an object as JSON, make sure to set isJson as true.

JsRender Template engine Helpers

- ~getResourceUrl(relativeModuleUrl)
  Returns a URL which can be embedded in HTML as a resource linked to a file in the cartridge package.
  For instance `<img src='{{:~getResourceUrl("icon.png")}}'/>`
- ~getServerUrl(relativeServerUrl)
  Returns a URL following a path in the connected Jive server.
  For instance `<a target="_blank" href='{{:~getServerUrl("/activity")}}'/>`

Debugging your cartridge

For code changes to take effect, you have to restart the browser or reload the extension (Chrome/Safari) each time you upload the cartridge via the Admin console.

The easiest way to debug is using Chrome or Internet Explorer. In both cases you can add the "*debugger;*" statement in order to break into debugger when approaching the statement. You can also use *console.log(object)* to print output into the debugger console.

Debugging Internet Explorer

Press F12 to open the developer tools, click the Script tab and "start debugging".

Debugging Chrome

1. To debug pagescript.js - press F12 to open the developer tools, then click the Sources tab.
2. To debug module.js or other background context – go to Extensions in the Settings page of Chrome, make sure "Developer mode" is checked, and click the URL near "Inspect views", then click the Sources tab.

Debugging Safari

Open the preferences dialogue, click Advanced and make sure “Show Develop menu in menu bar” is checked.

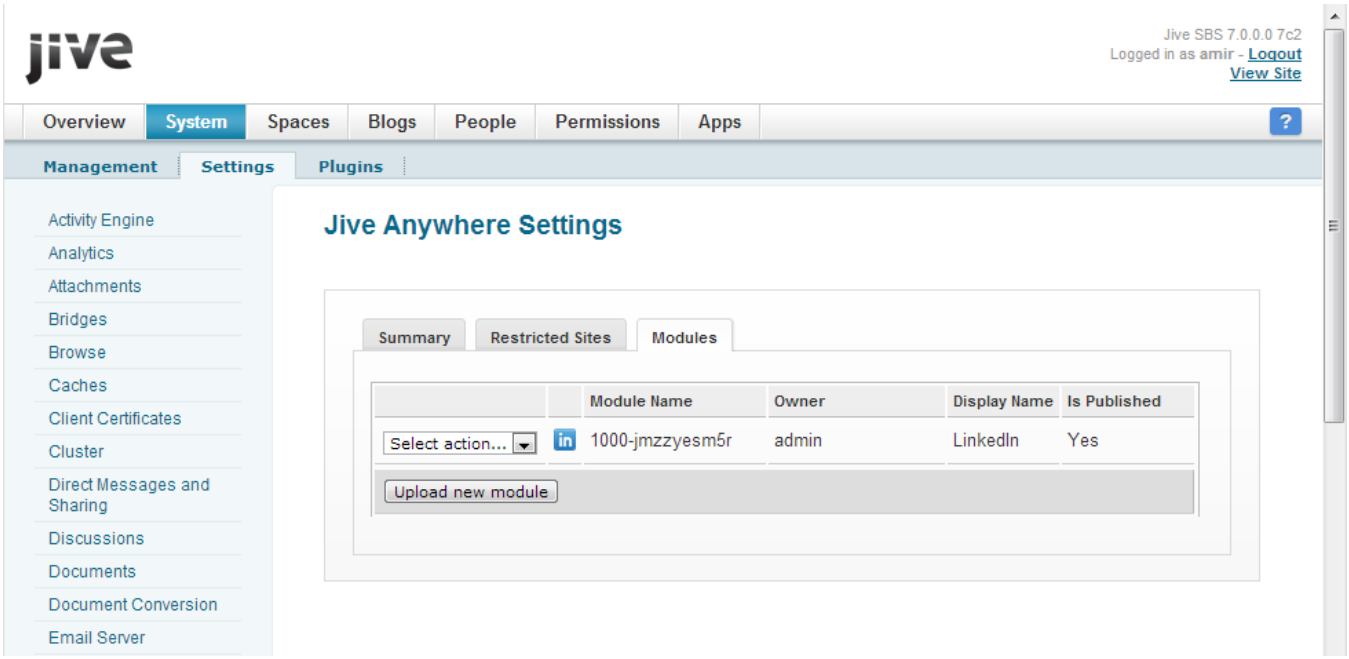
1. To debug pagescript.js - click the Develop menu and select “Start debugging JavaScript”.
2. To debug module.js or other background context – click the Develop menu and select “Show Extension Builder”, then click the “Inspect Global Page” button in Jive Anywhere.

Debugging Firefox

Due to the lack of ability to debug background extension context it is not recommended to use Firefox for debugging your cartridges. If no other option is available try using JavaScript alerts.

Manage cartridges via Admin console

Jive Anywhere cartridges are managed in the Admin console of Jive SBS. Go to **http://*your-jive-address*/admin** to enter the Admin console, then select **System -> Settings -> Jive Anywhere** at the menus and click the **Modules** tab. You must have an account with administrator privileges to enter this page.



Upload a new module

To upload a new cartridge, click **Upload new module** in the Modules tab.

In the displayed dialog enter the following fields:

- Type a display name for the cartridge.
- Choose the zip file of the new cartridge. The zip file must contain module.js and the optional files pagescript.js and icon.png as described in the Developer Guide section.
- Apply the cartridge included patterns. The cartridge will be activated only if one or more of the patterns matches the URL of the visited page. For instance *google.com* will match any web page in the google.com domain. You can also use Regular Expression patterns by prefix the pattern with a **$** sign. For instance *$[http|https]://google.com* will match either http://google.com or https://google.com.
  Type each URL pattern in the textbox field and click the "Up" arrow to apply. Select an existing URL from the list and click the "Down" arrow to edit. Note: only one cartridge can be applied for a web page. If more than one cartridge matches the current web page, only the first one will be applied.

Click the Submit button to upload the cartridge as Unpublished. While a cartridge is unpublished only its owner will be able to use it, in this case the Admin. After testing the cartridge you may choose to publish it.

Edit existing cartridge

To modify a cartridge click the "Select action…" dropdown list near the cartridge you wish to modify and select **Edit**. The Edit cartridge dialog will appear.

Remove existing cartridge

To remove a cartridge click the "Select action…" dropdown list near the cartridge you wish to remove and select **Delete**.

Publish and Un-publish a cartridge

After a cartridge has been tested it can be published to all users. Click the "Select action…" dropdown list near the cartridge you wish to publish or un-publish and select **Publish** or **Unpublish**.

Un-published cartridges will be only usable by their owners, while published cartridges will be usable by all users. All cartridges should be tested before being published.