

Rocket.Chat Integration

Version 1.4

Rocket.Chat Plugin

Overview

The Rocket.Chat plugin provides integration between Rock RMS and a Rocket.Chat server, allowing you to increase your community engagement with fully integrated chat functionality. The plugin only supports linking a single Rocket.Chat server.

Features

The Rocket.Chat plugin provides the following features:

- **Single Sign-On** – Allows your Rock login to be used for Rocket.Chat.
- **Rocket.Chat Account Provisioning** – Automatically create Rocket.Chat user accounts for your Rock RMS community members.
- **Group to Private Channel Sync** – This allows a group in Rock to automatically create a private channel and sync its members to it.
- **Profile Sync** – This feature allows profile information in Rock to be synced to Rocket Chat. This information includes:
 - Person Photo
 - Email Address
 - Rocket.Chat role assignments based on Rock data views, which enables badges to be displayed in Rocket.Chat
- **Chatbot** (think Chip in the Rock Community) – Chatbot functionality allows Rock to automate replies in Rocket.Chat and add some personalization features.
 - Responds to chat message triggers that you can configure in Rock
 - Gives out “chat points” based on community member feedback
- **Chat History** – Log Rocket.Chat messages to Rock interactions.
- **Mobile Access** – This allows individuals to get their Rocket.Chat account information for use by the Rocket.Chat mobile application.

Configuration Steps

These instructions assume you have already installed and configured a Rocket.Chat instance with an administrator account. If you haven't already done that, you'll want to start at <https://rocket.chat/install> and find the documentation for your preferred server or

cloud hosting provider. For the sake of these instructions, we will assume you have Rocket.Chat running at <https://chat.rockrmsdemo.com/> and your Rock website is <https://www.rockrmsdemo.com/>. Any time you see those URLs in the instructions below, you should replace them with your server's address.

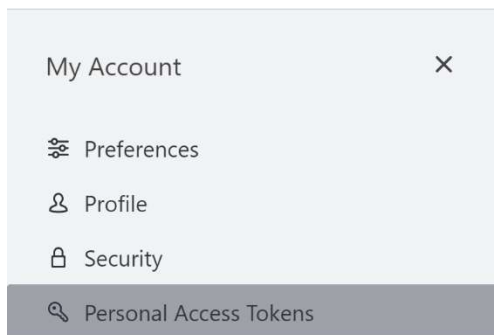
Step 1: Install the Rocket.Chat plugin from Rock Shop

Log in to the Rock Shop and download the Rocket.Chat plugin. After installation two pages will be added to your Rock installation. These pages can be found under 'Admin Tools > Installed Plugins'.

Step 2: Create a Personal Access Token in Rocket.Chat

This step generates the API credentials Rock will use to access your Rocket.Chat server.

1. Log in to your Rocket.Chat server with an administrator account. Please note that the account you choose will be the account that Rock uses for making the API calls to support the integration. You may want to consider creating a special account for this purpose.
2. Click on your user avatar in the top left corner, and then click "My Account".
3. Select "Personal Access Tokens" from the "My Account" menu.



4. Create a name for your token (something like "Rock RMS") and click the "Add" button.

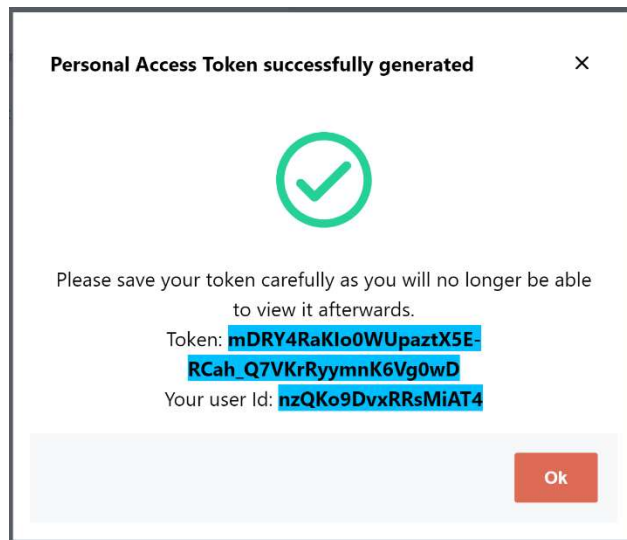
Personal Access Tokens

Personal access tokens to REST API

Rock RMS

Add

5. Rocket.Chat will give you two strings, a Token and a user Id. You will need both values for the next step.



Step 3: Configure the Rocket.Chat plugin settings

This step tells Rock what it needs to know to communicate with your Rocket.Chat server, using the credentials we created in Step 2.

1. Copy the Personal Access Token values from Step 2 into the Rocket.Chat settings.
2. Log in to your Rock admin portal (e.g., <https://www.rockrmsdemo.com/admin>).
3. Open the "Admin Tools" menu and click "Installed Plugins".
4. Click on "Rocket.Chat Settings" and enter the correct configuration values from Step 2:
 - Rocket.Chat Server URL – Enter the URL of your Rocket.Chat server.
 - Rock API Username – Rocket.Chat User Id.
 - Rock API Password – Rocket.Chat Token Password.
 - Allow User Self-Registration – If this option is not enabled, users will be unable to access Rocket.Chat until an account has been created for them by the data synchronization job (which is explained in more detail later in this document). This option is enabled by default.

Rocket.Chat Settings

[Home](#) > [Installed Plugins](#) > Rocket.Chat Settings

Integration Settings for your Rocket.Chat instance.

Rocket.Chat Settings

Rocket.Chat Server URL

API Username

API Password

Self-Registration

☒ Allow User Self-Registration

Save

Step 4: Enable the appropriate CORS policy

In this step, we will configure the Cross-Origin Resource Sharing (CORS) policy which tells Rock that it's okay to allow access to Rock resources from Rocket.Chat. This is necessary to prevent Rock's built-in security from blocking access to the REST endpoint that Rocket.Chat will use to authenticate users.

1. Open the "Admin Tools > Security > REST CORS Domains".
2. Click on the Add button and enter the Rocket.Chat Server URL in the Value (e.g., <https://chat.rockrmsdemo.com>). Enter a description (if you want to) and click the "Save" button.

REST CORS Domains

[Home](#) > [Security](#) > REST CORS Domains

REST CORS Domains (Advanced)

▼

Lists the external domains that are authorized to access the REST API through "cross-origin resource sharing" (CORS).

REST CORS Domains (Advanced)

⌵

⌵

+

Value	Description
⌵ https://chat.rockrmsdemo.com	Rocket.Chat

50

500

5,000

1 Defined Value

⌵

⌵

+

WARNING: Azure App Service implements its own CORS controls, and you will also need to enable Access-Control-Allow-Credentials and add your Rocket.Chat Server URL as an "Allowed Origin" in the Azure App Service, under API > CORS.

Please be aware that other hosting providers, particularly those that implement load balancers or other forms of reverse proxy servers may have similar configuration requirements.

Step 5: Enable the allowed frame domain

In this step, we'll tell Rock to permit the Rocket.Chat server to load Rock pages within a frame. This enables the Rocket.Chat server to send users to the Rock login form if they're not already authenticated.

1. Open the "Admin Tools > CMS Configuration > Sites".
2. Select the site you plan to use for your audience to use to login into Rock for access to Rocket.Chat, and then click on the "Edit" button.
3. Expand the "Advanced Settings" section at the bottom.
4. Edit the "Allowed Frame Domain(s)" setting to include the Rocket.Chat server.

- If this value is blank, remember to change the domain name to your own Rocket.Chat server, and enter:

'self' chat.rockrmsdemo.com

Please Note: The 'self' entry is required and if you leave it out, you will break Rock's modal popups. Rock includes this value by default when you don't have something else in this box. You can also explicitly enter the domain of your Rock server, here, but it's better to simply let the browser figure it out (by specifying 'self'). Be sure to not include the https://.

- There may already be a value configured, here, if your site has been configured to allow frames for another domain. If there is already a value present, add a space and the Rocket.Chat server domain.

Allowed Frame Domain(s) 

Step 6: Enable Iframe Authentication in Rocket.Chat

This step configures Rocket.Chat to send users who are not already authenticated to the login form in Rock, using an iframe.

1. Click on the "Options" menu in the top left corner (the menu icon with three vertical dots).
2. Click "Administration".
3. Scroll down on the menu on the left and click on "Accounts" under the "Settings" header.
4. Expand the "Iframe" section and enter the appropriate configuration values for your server:
 - a. Enabled: True
 - b. Iframe URL: <https://www.rockrmsdemo.com/RocketChatLogin>
 - c. API URL: <https://www.rockrmsdemo.com/api/RocketChat/Sso>
 - d. Api Method: GET

Accounts

CancelSave changes

Iframe

^

Enabled

☒ True ☐ False

Iframe URL

API URL

Api Method

Reset Section Settings

Reset

5. Select the "Save Changes" at the top of the page.

Step 7: Disable the Rocket.Chat API Rate Limiter

This step is necessary to prevent the Rocket.Chat plugin from hitting rate limits that will prevent it from accomplishing tasks it needs to do.

1. Click on the "Options" menu in the top left corner (the menu icon with three vertical dots).
2. Click "Administration".
3. Scroll down on the menu on the left and click on "Rate Limiter" under the "Settings" header.
4. Expand the "API Rate Limiter" section.
5. Change the value of "Enable Rate Limiter" to "False".

API Rate Limiter

Enable Rate Limiter ☐ True ☒ False

Chatbot Setup

The Chatbot allows Rock to add automation to your Rocket.Chat community. These features include:

- **Points** – Rock can listen for certain patterns to give community members points. These patterns look like: @teddecker ++ or @teddecker - -
- **Automated Responses** – Rock can also look for regex patterns in messages and automate a canned reply. More on setting this up below.
- **Message Logging** – The integration can also log messages to Interactions in Rock.

Configuring Rocket.Chat to Use the Rock Webhook

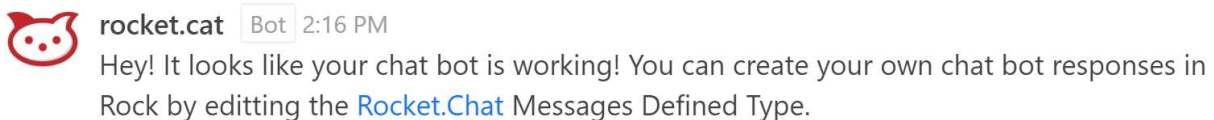
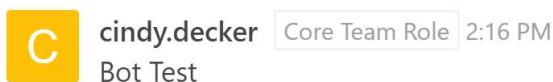
In this step, we'll connect Rocket.Chat to the Webhook in Rock that enables the chat bot features (including automated responses, points, and chat history within Rock). The Rock Webhooks is already set up by the plugin installation process, so this step is done entirely within Rocket.Chat.

1. Choose or create a Rocket.Chat user for your integration. This user should be a member of the 'bot' role. You can create this user under the "Users" tab of the Rocket.Chat administration options. One named 'Rocket.Cat' comes with your Rocket.Chat install.
2. Select "Integrations" from the "Administration" menu.
3. Click the "New Integration" button in the top right corner.
4. Select "Outgoing Webhook" and enter the following values:

- Select "Message Sent" as the Event Trigger.
- Set "Enabled" to "True". **This is important, so make sure you don't skip this step as the value is defaulted to 'False'.**
- Enter a name for your integration (we recommend something like "Rock Integration",
- Enter "all_public_channels,all_private_groups" as the Channel value. This informs Rocket.Chat where the bot should be active, and these values mean it will operate in public channels and private groups (but not direct messages).
- Enter "https://www.rockrmsdemo.com/Webhooks/RocketChatMessages.ashx" in the URLs box. You didn't forget to replace that domain name with your own domain, right?
- Put the desired Rocket.Chat user's name in the "Post as" box.
- Leave the rest of the values on their default settings and click "Save changes" in the top right corner.

Configuring Automated Chat Triggers

The Rocket.Chat bot is configured to respond to only one trigger, by default, and that's "Bot Test", like this:



This is the same code we use for Chip, though, and you can configure your own responses to any queue you want by adding Defined Values to the Rocket.Chat Messages defined type (Admin Tools > General Settings > Defined Types > Rocket.Chat Messages).

The basic format is that the "trigger" is the value, and the response is the description. You can use regular expressions in the trigger to make it match patterns. You can also supply multiple potential responses in the description by putting each response on a different line and the chat bot will select one at random. Please note that this means your responses must be on a single line. If you put in multiple lines, they will be treated as separate responses and the user will only receive one of them.

Rocket.Chat Sync Job

The provided sync job adds additional capabilities to the integration. These include:

- **Profile Sync** – This provides a one-way sync of Rock profile photos and emails to Rocket.Chat.
- **Group Sync** – This feature allows you to sync a group in Rock to a provide group in Rocket.Chat. More details on the settings of this feature are below.
- **Role Sync** – This allows you to provide data views in Rock that will be synced to roles in Rocket.Chat.

Some of those require some additional setup, but don't worry, we'll get to that in a minute.

1. Open the "Admin Tools > System Settings > Jobs Administration" in Rock.
2. Click on the "Add" button to create a new job.
3. Choose a name for the job (like "Rocket.Chat Sync").
4. Enter a Cron Expression to determine how often the job should run.
5. Select "org.sparkdevnetwork.RocketChat.Jobs.SynRocketChat" as the Job Type.

Job Configuration Options

This job type includes some custom configuration options which change the behavior of the job.

Job Type •

org.sparkdevnetwork.RocketChat.Jobs.SynRocketChat ▼

Resync Passwords i

No ▼


Disable Accounts Without Groups i

Yes ▼

Delete Inactive Groups i

Yes ▼

Exclusion Group i

 Banned From Rocke... ▼

Save Cancel

Resync Passwords – This option will synchronize passwords between Rocket.Chat and the one stored in Rock, by updating the Rocket.Chat password. Please note that using this option will cause mobile sessions to expire, and the user may need to get their password back from within Rock.

Sync Photos – This option will synchronize Rock photos to Rocket.Chat avatars. This option is enabled by default.

Sync Email – This option will synchronize email addresses from Rock to Rocket.Chat. This option is enabled by default.

Disable Accounts Without Groups – This option treats any user who is not a member of a “Rocket.Chat Channel” group (which is explained in further detail below) as an inactive user and disables their Rocket.Chat account. This means that those users will be blocked from accessing Rocket.Chat.

Delete Inactive Groups – This option will remove any groups from your Rocket.Chat server which are not associated with a “Rocket.Chat Channel” Rock group or which no longer have any active members. This setting is enabled by default. You may disable this setting if you prefer to leave this groups on our Rocket.Chat server.

Exclusion Group – This option allows you to select a group and any members of that group will have their Rocket.Chat account disabled.

Configure Groups for Synchronization

Rock Groups are not automatically synchronized to Rocket.Chat. To enable groups to be synchronized, you will need to create a new Group Attribute with a Key of RocketChatChannel (this key must match exactly, so make sure you type it correctly!) and a Boolean Field Type.

You can create as many attributes with this key as you want, in case you want to allow multiple Group Types to become Rocket.Chat Channels. You can set these attributes up by going to Admin Tools > General Settings > Group Types.

Here's an example of how this attribute might be set up:

The screenshot shows a web form titled "Group Attributes" with a sub-header "Add attribute for groups of group type General Group". The form contains several fields and options:

- Name:** A text input field containing "Rocket.Chat Channel".
- Active:** A toggle switch labeled "Yes" with a blue information icon.
- Abbreviated Name:** An empty text input field.
- Description:** A large empty text area.
- Categories:** A dropdown menu showing a folder icon.
- Field Type:** A dropdown menu set to "Boolean".
- Key:** A text input field containing "RocketChatChannel".
- True Text:** A text input field containing "Yes".
- False Text:** A text input field containing "No".
- Required:** A checkbox labeled "Require a value" which is unchecked.
- Show in Grid:** A checkbox labeled "Yes" which is unchecked.
- Default Value:** A dropdown menu.
- Advanced Settings:** A collapsed section at the bottom.

At the bottom right of the form are "Save" and "Cancel" buttons.

NOTE: The Group Attribute Key is "RocketChatChannel" and does not contain a period.

You may also want to edit the security of this attribute to determine who should be allowed to view/edit it. By default, only administrators will be able to view the attribute.

Once you've set the Entity Attribute up correctly, you should see it on groups that match the Group Type you selected (assuming that was the criteria you used to define the attribute, of course). This value must be set to true for the Rocket.Chat Sync Job to recognize the group.

Group Attribute Values

Topic 

Book of Genesis

Rocket.Chat Channel 

Yes

The Rocket.Chat synchronization job will create private group chat rooms in Rocket.Chat for any groups in Rock which you have set with the "Rocket.Chat Channel" attribute, and it will add and remove users from that channel to mirror the membership in Rock. Additionally, if a member is marked as a leader of the group in Rock, they will be marked as a leader *and* a moderator in Rocket.Chat.

Data Views to Rocket.Chat Role Sync

NOTE: The Rocket.Chat plugin does not automatically create these roles in Rocket.Chat. You must create the matching role there manually before it will synchronize.

You can assign Rocket.Chat roles by creating data views that include Rocket.Chat users. The name of the data view needs to match the Rocket.Chat role exactly, and you will need to create the role on your Rocket.Chat server manually (Administration > Permissions).

There are several ways you can utilize this functionality, but one way it can be used is to create custom badges for your users by adding custom CSS that modifies the role tag. For example, below are the CSS rules we use to create the Core Team badge in the Rock community chat. This is added in Rocket.Chat under 'Administration > Layout > Custom CSS'.

```
.role-tag[data-role="Core Team"] {  
    padding: 1px 8px 1px 0;  
    color: #ee7725;  
    border: 0  
}  
  
.role-tag[data-role="Core Team"]::before {  
    content: "";  
    background-image: url("https://chat.rockrms.com/emoji-custom/Rock.png");  
}
```

The result looks like this:



To accomplish this, you will create a data view in the "Rocket.Chat Roles" data view category, which was added when you installed the plugin. Here's a simple example to match the badge above.

Core Team


Id: 15 + Create Report


Core Team Badge Dataview

Applies To Person	Filter In groups: Core Team
Category Rocket.Chat Roles	

Edit

Delete





Additional Technical Details

New Blocks and Pages

The Rocket.Chat plugin installs the following new blocks and pages.

Blocks:

- **Rocket.Chat Settings Block** – This block allows you to enter the Rocket.Chat integration settings (server URL, API username and password).
- **Rocket.Chat Login Block** – This block utilizes Lava to redirect users to Rocket.Chat after ensuring that they are logged in. You can customize the behavior of this block by editing the Lava.
- **Rocket.Chat Intro Block** – This block displays some information about Rocket.Chat to your users, including their Rocket.Chat username and password which they will need to access the mobile chat applications. It also includes download links for those applications. This block also utilizes Lava, so you can customize how the page appears.

Pages:

- **Rocket.Chat Settings Page** – This page uses the Rocket.Chat Setting block and is installed in the “Installed Plugins” area of your Rock RMS site.
- **Rocket.Chat Login Page** – This page uses the Rocket.Chat Login block and is installed at the /RocketChatLogin page route.

- **Rocket.Chat Intro Page** – This page uses the Rocket.Chat Intro Block and is installed at the /RocketChat page route.

Displaying A User's Rocket.Chat Credentials

The included Rocket.Chat Intro page will display a user's credentials and instructions for downloading the Rocket.Chat mobile applications. That page utilizes a block which contains customizable Lava, so you can edit it to meet your needs, but you can also accomplish this with some custom Lava.

The example below shows how you can display a user's credentials with Lava.

```
<div class=""panel panel-default"">
  <div class=""panel-heading"">
    <i class=""fa fa-rocketchat""></i> Rocket.Chat
  </div>
  <div class=""panel-body"">
    {% if CurrentPerson %}
      {% assign chatName = CurrentPerson |
Attribute: 'orgSparkDevNetwork_RocketChatUsername' | Trim %}
      {% if chatName and chatName != '' %}

        <div class=""form-group"">
          <label class=""control-label"" >Username</label>
          <div class=""control-wrapper"">
            <p class=""form-control-static"">{{ CurrentPerson |
Attribute: 'orgSparkDevNetwork_RocketChatUsername' }}</p>
          </div>
        </div>

        <div class=""form-group"">
          <label class=""control-label"">Password</label>
          <div class=""control-wrapper"">
            <p class=""form-control-static"">{{ CurrentPerson |
Attribute: 'orgSparkDevNetwork_RocketChatPassword' }}
</p>
          </div>
        </div>
      {% endif %}
    {% endif %}
  </div>
</div>
```

Rocket.Chat's Unique Email Address Requirement

Rocket.Chat requires each user to have a unique email address. This conflicts with Rock, where, for example, family members might share a single email address. To work around this limitation, Rock will assign an email alias, by adding "+rc" and a number after the username portion of their email address (before the @ symbol). For example, if two three people access your Rocket.Chat server who all use the email address "chip@sparkdevnetwork.org", they would receive the following email address assignments:

- First User: chip@sparkdevnetwork.org
- Second User: chip+rc1@sparkdevnetwork.org
- Third User: chip+rc2@sparkdevnetwork.org

You should understand that this may make email delivery through Rocket.Chat somewhat unreliable, as not all email systems permit this method of aliasing.

Rocket.Chat's Unique Username Requirement

Rocket.Chat also requires a unique username for each user. To make it as simple as possible to sign up, we automatically provision a username for them, by taking the full name and replacing spaces with a period. So, "Rock Lobster" would become "rock.lobster" in Rocket.Chat. Since it's possible that different people might have the same name, we also append a number to the username if it already exists (i.e., "rock.lobster2", and so on).

User Avatar URLs

Rock builds the URL for your user's avatars from the "Public Application Root" setting. That site needs to be accessible to supply your user's avatars. You've probably already configured this, or your website wouldn't work, but if you need to configure it, you'll find it under General Settings > Global Attributes.

Rocket.Chat Log

On the Rocket.Chat Settings page (Admin Tools > Installed Plugins > Rocket.Chat Settings) you will find a button to view the Rocket.Chat log. This log will report any errors that occur when attempting to communicate with the Rocket.Chat API as well as every attempt to create a new user account in Rocket.Chat.

Failed Access Attempts

If the Rocket.Chat plugin is unable to log a user in or automatically create their account for them, it will log an error to the Rocket.Chat log and display that error to the user.

This behavior is necessary to prevent the user from being caught in an endless loop of failed login attempts. This might occur for a variety of reasons, like network problems or problems with the Rocket.Chat server, but the most common reason this might occur is due to a request to create an account which your Rocket.Chat server rejects. The most likely cause of this scenario is an email address which has been blacklisted.

Cross-Origin Resource Sharing (CORS) Policy

The Rocket.Chat integration uses a client-side AJAX request to get an authentication token from Rock. This request is issued through the user's browser, which will not allow requests like this by default, because it can be a security risk when it happens across sites that can't be trusted. You may have noticed that we added the domain of the Rocket.Chat server to Rock's list of approved CORS REST domains (in Step 4). This adds an HTTP header that informs the browser that the request should be allowed when it comes from your Rocket.Chat server. This header is necessary for the integration to function.

Changes made to Rock when you install the Rocket.Chat plugin

We've discussed some of these in the configuration instructions, but here's the technical detail of what the plugin adds to your Rock system.

Person Attributes

The Rocket.Chat plugin creates a new category of Person Attributes, named "Community Config". It creates the following attributes within that category:

- Rocket.Chat Username
 - Holds the user's Rocket.Chat Username.
- Rocket.Chat Password
 - Holds the user's Rocket.Chat Password. This value is randomly assigned by Rock, and you will need to expose it to the user to permit them to access the mobile chat applications.

- Rocket.Chat Id
 - This is the unique identifier for the user, assigned by Rocket.Chat when the account is created.
- Chat Points
 - Tracks how many points the user has received from the chat bot. This attribute is not utilized if the chat bot is not configured.
- Rocket.Chat Last Avatar Update
 - This is a DateTime value used to track when the avatar was last updated in Rocket.Chat so that the synchronization job knows if it needs to be updated.
- Rocket.Chat Account Seen Inactive
 - This is a DateTime attribute that tracks failed login attempts due to the Rocket.Chat account having been disabled. This prevents Rocket.Chat from sending the user through a login loop and generating repetitive requests for the login service. Please note that once an account has attempted to log in after being marked inactive, it will take two hours before this flag is reset, and the user cannot login (through SSO) before then.
- Rocket.Chat Login Error
 - This value temporarily tracks failed attempts to log a user in to prevent them from getting stuck in an endless loop when an attempt to log a user in or create their account in the Rocket.Chat API fails. It gets reset immediately after being assigned, so you shouldn't expect to see any information, here, but you will find the error in the Rocket.Chat log. See the section about "Failed Access Attempts" under the "Additional Technical Details" heading, above, for more information about this.

You can, of course, access these attributes the same way you would with any others, including through Lava.

Group Attribute

The Rocket.Chat plugin adds a new Group Attribute to all groups named "Rocket.Chat Channel". This attribute is a Boolean value which informs the Rocket.Chat synchronization job that this group should be synchronized with Rocket.Chat. This attribute has no effect if that job is not enabled.

Data View Category

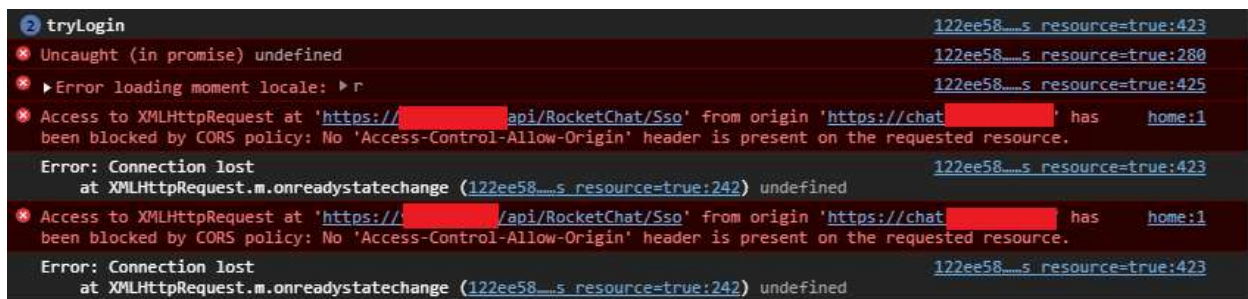
The Rocket.Chat plugin adds a new Data View Category named “Rocket.Chat Roles”. For details on what this category does, see configuration step 8b.

Troubleshooting Common Issues

Post-Login Loop

If your Rocket.Chat integration configuration is not correct, you may experience a loop, where your browser constantly attempts to refresh after you log in to Rock. If this occurs in your environment, you can monitor the requests using your browser's integrated developer tools (which you can open by pressing F12 in Google Chrome or Microsoft Edge). Checking the “console” tab of the developer tools will typically show you what errors are occurring.

This typically occurs when the CORS policy settings are incorrect, in which case you will see errors like the ones below.



The first thing to do in this scenario is double-check that you have entered the correct configuration data from Step 4 (the CORS REST domain value should be the same URL that your browser is trying to access when the above errors are occurring) and Step 6 (the API URL must be correct and the API method must be “GET”). You can verify that the API URL is functioning by accessing it in your browser after you are logged in to Rock.

If you have verified the above information and restarted your Rock service, and you are still experiencing the error above, you will need to look at other pieces of your infrastructure which may have control over your CORS headers. In particular, if you are using Azure App Service, you will need to make additional configuration changes within your Azure Control Panel (please refer to the warning note in Step 4 for more information).