ReMi – Basic User Guide

Contents

[1. Introduction 3](#_Toc432757308)

[2. Logging In 3](#_Toc432757309)

[2.1. Create Account 3](#_Toc432757310)

[2.2. User Profile 3](#_Toc432757311)

[3. Emails are sent via email plugin, so if this has not been configured, notifications will not be sent.Scheduling Releases 4](#_Toc432757312)

[3.1. Scheduling a Single Package Release 4](#_Toc432757313)

[3.2. Scheduling a System Maintenance Outage 5](#_Toc432757314)

[4. Release Participation 6](#_Toc432757315)

[4.1. Release Plan 6](#_Toc432757316)

[4.1.1. Release Participants 6](#_Toc432757317)

[4.1.2. Release Approvers 7](#_Toc432757318)

[4.1.3. Signing off on Releases 7](#_Toc432757319)

[4.1.4. Release Tasks 7](#_Toc432757320)

[4.1.5. Check List 9](#_Toc432757321)

[4.1.6. Release Jobs 10](#_Toc432757322)

[4.1.7. Release Content 10](#_Toc432757323)

[4.1.8. Release Changes 11](#_Toc432757324)

[4.1.9. Release Decision 11](#_Toc432757325)

[4.2. Release Execution 11](#_Toc432757326)

[4.2.1. Release Tasks 12](#_Toc432757327)

[4.2.2. Release Process (metrics) 12](#_Toc432757328)

[4.2.3. Signing off and closing release 12](#_Toc432757329)

[4.2.4. Release Jobs Measurements 12](#_Toc432757330)

[4.3. Other release types 12](#_Toc432757331)

[4.3.1. Hotfix 13](#_Toc432757332)

[4.3.2. Request for Change 14](#_Toc432757333)

[4.3.3. Automated Release 14](#_Toc432757334)

[4.3.4. System Maintenance, PCI, 3rd Party and Corp IT 15](#_Toc432757335)

[5. Basic plugins configuration 15](#_Toc432757336)

[5.1. Global plugins configuration 15](#_Toc432757337)

[5.2. Package plugins configuration 16](#_Toc432757338)

[5.3. Jira plugin 16](#_Toc432757339)

[5.4. Jenkins plugin 18](#_Toc432757340)

[5.5. GO plugin 19](#_Toc432757341)

[5.6. Gerrit plugin 19](#_Toc432757342)

[5.7. ZenDesk plugin 21](#_Toc432757343)

# Introduction

This guide details how to access and use the ReMi functionality. Topics that will be covered include user registration and profiles, scheduling/planning releases, release approval, execution and sign off; as well as configuration for supported external system plugins.

# Logging In

## Create Account

ReMi uses the authentication plugin to authenticate users and create user accounts upon the first login. The basic plugin uses LDAP authentication to create an account and assign the user to the restricted ‘Basic User’ role when the user logs in using their LDAP credentials. The ReMi Administrator may change the user role allocation from ‘Basic User’ to an appropriate role that will allow the user to access the release planning and execution functionality they require.

## User Profile

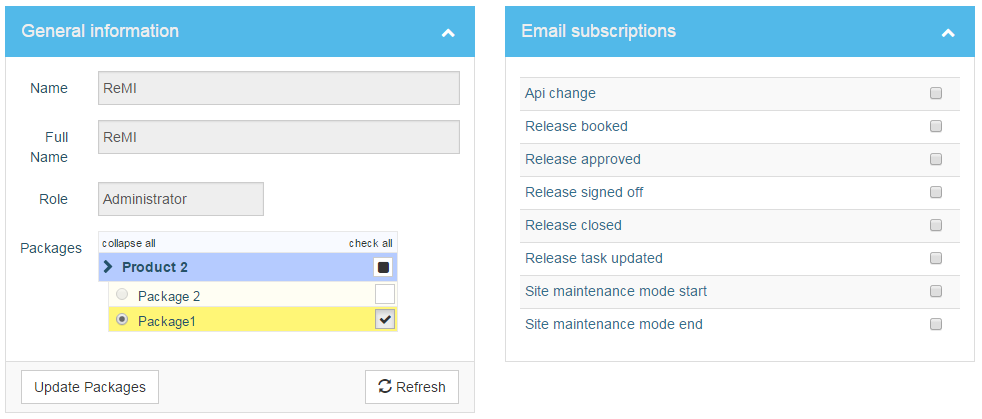
Each account on ReMi has a profile that stores information about what packages the user needs to have access to for release planning and execution as well as email notification preferences.

To access your profile, click on your screen name displayed in the top right navigation bar as per the screen shot below.



Depending upon your user role you will be able to allocate packages to the profile (as per the screenshot below) that you wish to interact with. In the event that you are not in a role that has access to this functionality (the ‘Update Packages’ button is disabled) you will need to request access from a ReMi Administrator. ReMi does not automatically subscribe users to email notifications, so in order to be notified of events that you are interested in you will need to check the boxes for each event you want to receive notifications about. The types of notifications available for subscription are:

* API changed - a new version of ReMi was released and there where changes to ReMi API
* Release booked - new release was scheduled in the calendar
* Release approved - all approvers signed off on the release go ahead
* Release signed off - release concluded successfully and all signatories signed off on it.
* Release closed -The release has been closed out and the Release Notes are emailed to stakeholders
* Release task updated – a manual release task for release was created or updated.
* Site maintenance mode start - Site was set offline to start deployment
* Site maintenance mode end - Site is back online again, after deploy is completed

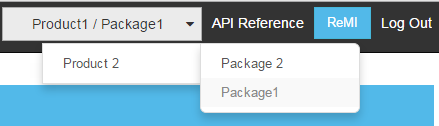


# Emails are sent via email plugin, so if this has not been configured, notifications will not be sent.Scheduling Releases

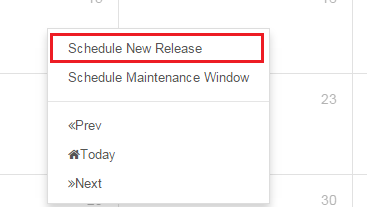
## Scheduling a Single Package Release

The following steps need to be taken to schedule new release for a single package:

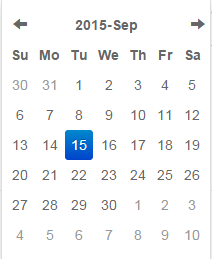
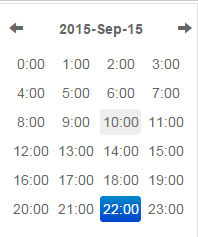
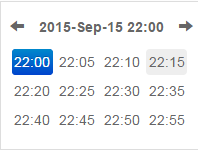
1. Open the ReMi home page to see the release calendar
2. From the dropdown menu on the navigation bar select the package you want to schedule a release for.



1. Right click on day you want to schedule the release.
2. From context menu select ‘Schedule New Release’



* In the resulting pop up form fill in the information about the release: Sprint – ReMi is designed to work with agile methodologists, where a release would be executed for each sprint cycle, this field also can be used as release short description in the event that another methodology is being followed.
* Start/End time – where time can be specified within a 5 minute precision. The DateTime picker executes in three steps: first select the Date, then the Hour and finally the Minute that the release will be scheduled for

* Release Type – There are 4 release types for single package release:
  + Scheduled Release – Regular release after sprint end.
  + Hotfix – Release to fix an urgent bug on the production environment which needs to be executed immediately.
  + Request For Change – For any change that needs to be effected in production, usually this doesn’t require down time to the running systems
  + Automated Release – this type is not shown on the list. Automated releases can be only created through the ReMi API.
* Downtime Required – indicates whether system down time is required for the release
* Description – more detailed description of the release, up to 512 characters are supported..

Scheduled releases can be updated while they are still in effect. To update a release ensure that the package context is selected from the package list in the navigation bar, click on the release on the calendar and edit the information as required in the resulting pop up form. Once a release changes are saved all the subscribers to the release update notification will receive an email notification.

Approved and signed off releases will be ‘cleared’ when updated, which means that the release plan will have to be reworked

## Scheduling a System Maintenance Outage

System maintenance can affect one or more deployed systems. The most common system maintenance tasks are related to system/software updates, infrastructure changes, such as resizing hard disks or 3rd party tool provider outages which affect service availability in the production environment.

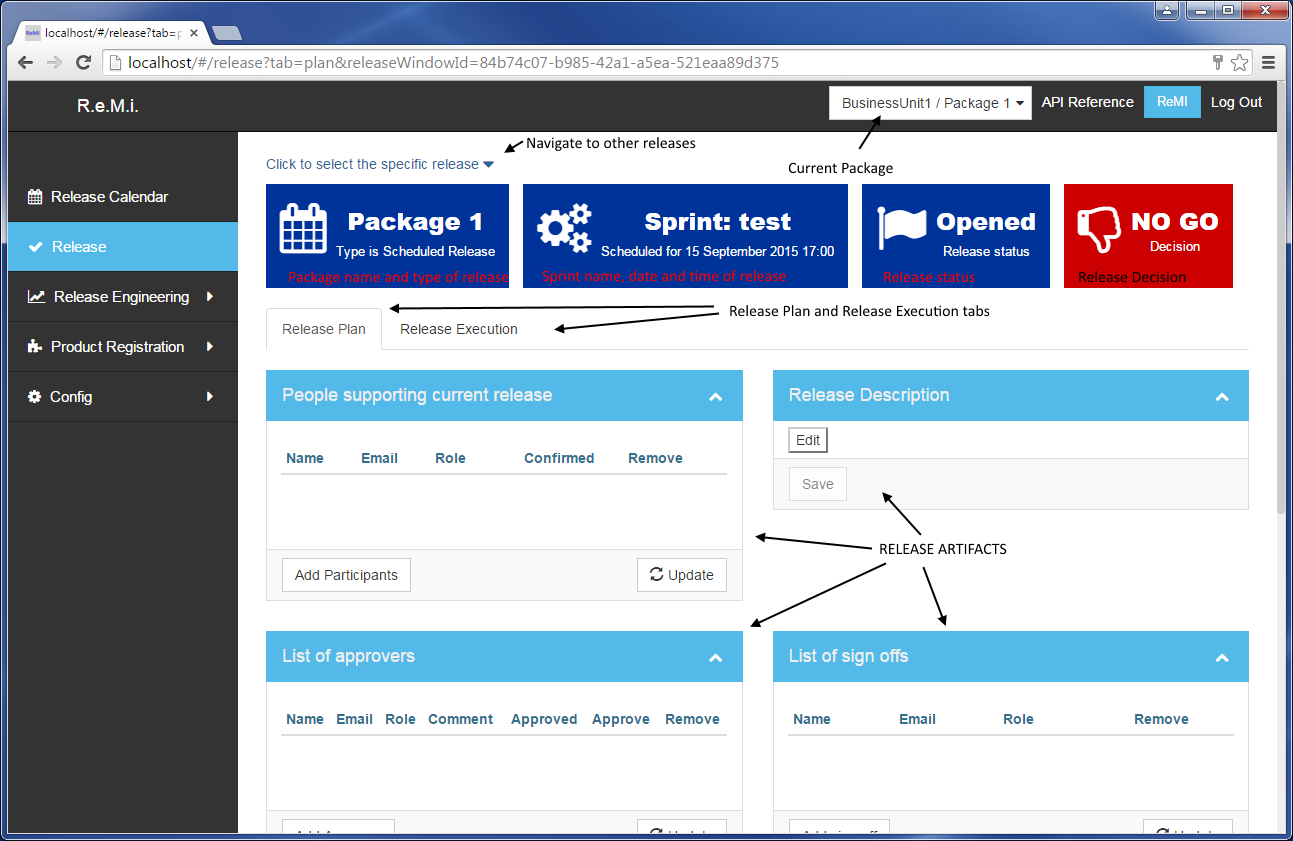
Scheduling a system maintenance window is very similar to scheduling regular release and can be created from the release calendar by right clicking the appropriate day and selecting the ‘Schedule System Maintenance’ option. The following information about the outage must be filled out on the ‘New System Maintenance’ pop up form:

* Headline – first 10 characters of this text will be shown on the calendar
* Start/end – same as on regular release.
* Packages – Select the packages that will be affected by the maintenance outage.
* Type – there are for types of System Maintenance:
  + PCI – it’s release to PCI compliance
  + System Maintenance – any system changes that do not require a system release
  + Corp IT Infrastructure changes
  + 3rd party – Scheduled outage for 3rd party systems, which may affect the production environment
* Downtime required – if system maintenance requires the system to be taken offline.
* Description.

# Release Participation

There are 4 ways to navigate to the release page:

* Directly from the navigation bar on left hand side by clicking the ‘Release’ link. This will open the latest not closed release in scope for the package context you have selected in the top right navigation bar.
* Right click on the release label in the calendar and select the ‘Navigate to Plan’ option from the context menu.
* Left click the dropdown button next to release label in the calendar and select the ‘Navigate to Plan’ option from the context menu.
* Click release label in the release calendar and when the edit popup form loads, click the ‘Navigate to Plan’ button.



## Release Plan

### Release Participants

Participants are persons whose presence is required during release planning or execution. To add participants click on the ‘Add Participants’ button from the ‘People supporting current release’ box. This will open a modal window with which users can be added. The autocomplete box helps to filter users during search, and is activated once 4 characters are typed into the search field,

The search functionality allows you to select more than one participant per search and then add all the users as participants once they have been selected.

Every participant added to the release will receive a calendar invitation for release containing a link to the confirm participation page. Each release participant may also confirm participation from the ‘People supporting current release’ box, by clicking the ‘Confirm’ button next to their names.

### Release Approvers

The release plan should be approved and signed off before the scheduled execution for the release. To add an approver to the list, follow the same steps as adding participants. One or more release approvers may be added to the release plan.

Once satisfied to approve the release plan approvers will click the ‘Approve’ button next to their name and add any additional comment if necessary. Once all the approvals have been signed the release status will be changed to a ‘Go’ status from the default ‘No Go’ status. This flags the team that the release plan is approved and that they may proceed with the release as scheduled.

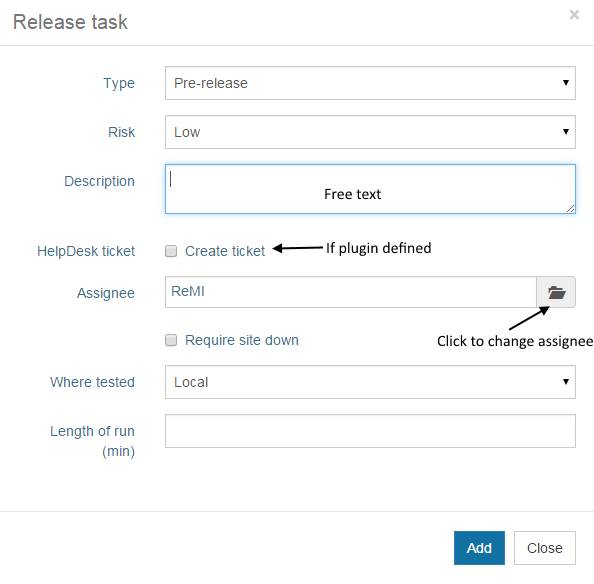
### Signing off on Releases

One or more stakeholders will be required to sign off and accept that the release has been completed as planned. The individual signing off on the release acknowledges that the release was executed and that the system was left in a functioning state post release.

### Release Tasks

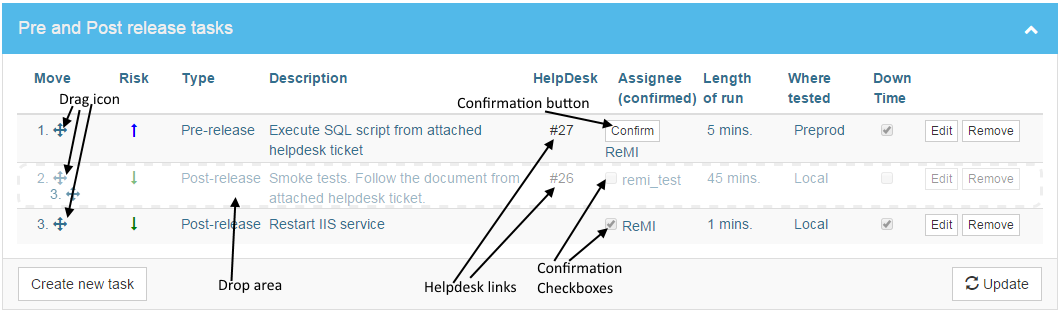
Tasks may be added to the release that may be executed before, during or after deploy is executed. A task is assigned to one user and acceptance of the task should be confirmed by that user before the release plan is approved. This confirmation can be done by clicking on link sent via the e-mail that is sent after assigning a user to the task, or by click ‘Confirm’ button next to user name for the release task.

To add a release task, click on ‘Create new task’ button and fill in the following fields:



* Type: – can be pre-release (starts before the release), post-release (starts after the release) or deployment task (not necessarily a pre or post deploy task).
* Risk: – can be low, medium and high.
* Description: – free text allowing up to 512 characters.
* Helpdesk ticket: – if the Helpdesk plugin is configured for the release package, this check box will be visible. Selecting this option will result in a ticket being created in the help desk system.
* Assignee – user who will execute the task. Searching for the user is very similar to searching participants, approvers and sign offs, with one minor difference, only one assignee can be selected for a task.
* Require site down: – if checked it means that task should be executed before or after deploy, while the site is still offline.
* Where tested – the task assignee will specify where the task was tested. For example if the task is about to run some SQL script on production database, it is advised to test on the preproduction environment.
* Length of run (min) – estimated time of runs of the task in minutes, to help better plan the release.

Release Tasks box



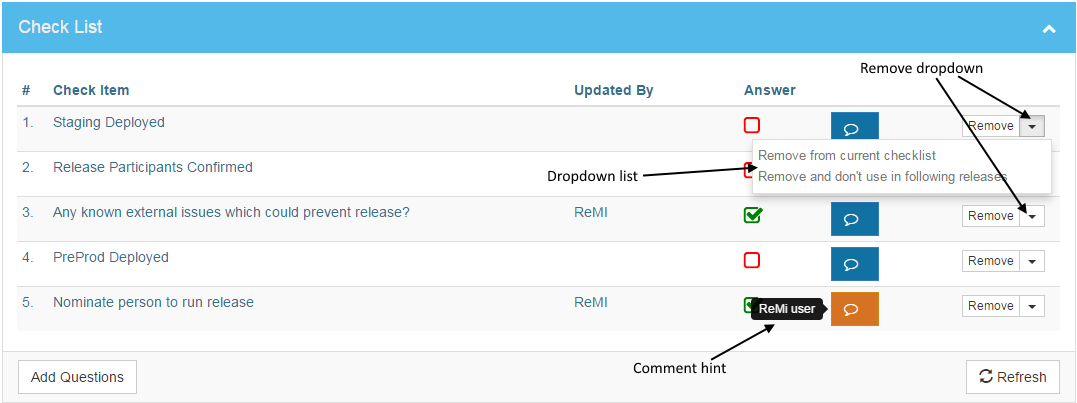
In the diagram above, you can see how to change the order of the tasks by dragging and dropping the task.

The diagram also shows the confirmation button and checkbox where the last task in the list is already confirmed.

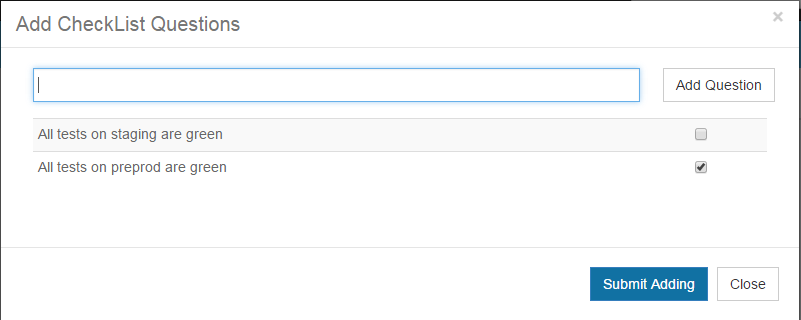
### Check List

The check list acts as a ‘pre-flight check’ to ensure that all the requirements to approve a release have been met. The check list is unique to each deployable package and can be modified as the situation dictates.

The diagram below demonstrates the usage of the checklist:



To add new item to the checklist click on ‘Add Question’ button, this will open the ‘Add CheckList Questions’ popup form. The auto-complete functionality will suggest previously added items that may be selected from the list and added to the current checklist.



### Release Jobs

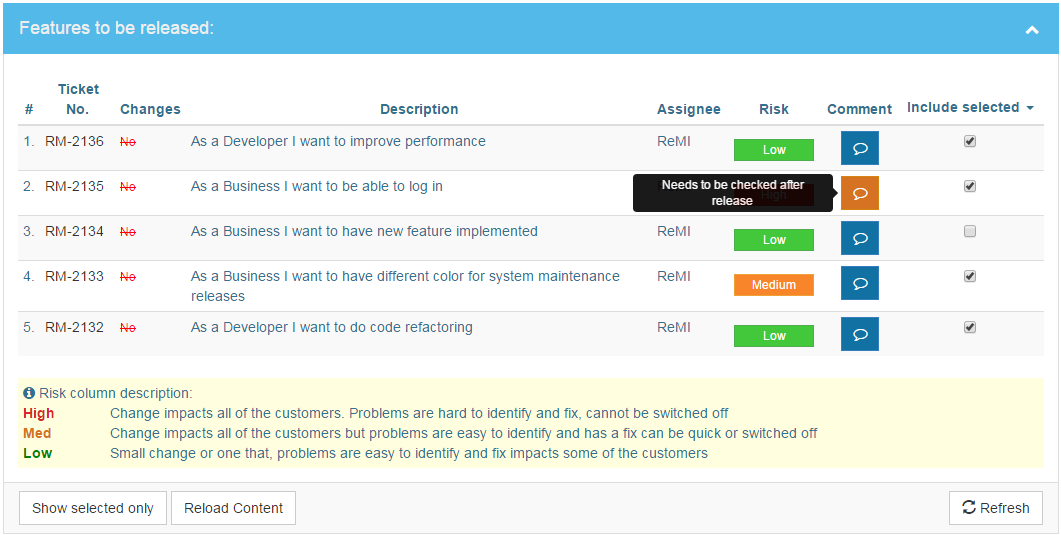
Release jobs are the deployment execution tasks and are populated from the deployment tool plugin configuration. For each release the jobs that will be run are checked so that only the required jobs are run during the release.

### Release Content

Release content consists of the features that will be released. This list is visible if the release content plugin has been configured to pull the content from the ticket tracking system (e.g. Jira). By default no content is selected so in order to include the features released in the automatically generated release notes, the items need to be selected from the release content list. For convenience there is an option to configure the releases so that all the content is selected by default. This can be done by configuring the package options from ‘Config/Business Units -> Package configuration’.

Any information associated to the tickets included in the release will only be stored in the ReMi system and will not be reflected in the ticket management system.

IMPORTANT! Release Content is stored in the ReMi database upon release approval.



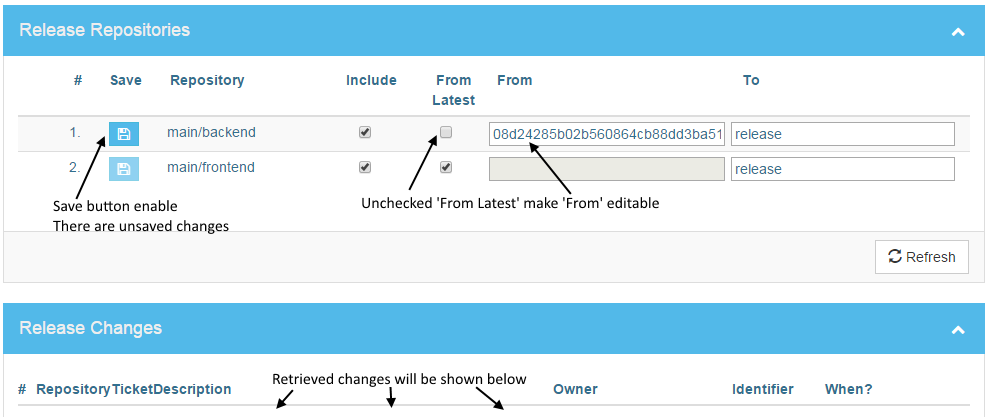
### Release Changes

Release changes refer to the code and configuration changes for the release, for example it can be list of git commits from repositories associated with the release package. To get release changes included in the release plan, the source control plugin for current package must be configured to fetch changes from the code repositories.

IMPORTANT! Release Changes are stored in ReMi database on release approval.

There are two ways of working with source control plugin:

* Changes are imported based on deploy jobs. For example the GO plugin will import changes from the configured pipeline jobs. Changes can be imported directly from repositories.



Similar to release jobs you can include or exclude repositories from release. The ‘From’ and ‘To’ fields operate similarly to the command ‘*git log from..to’*, so it can be used to fetch changes based on commit hash, tag name, branch name, short hash, etc.

The ‘From’ field may also be taken from the ReMi database if ‘From Latest’ is selected, if there is a latest commit stored in database, the hash of this commit will be shown in the ‘From’ column as read-only field, so if this input field is empty and read-only, you should uncheck ‘From Latest’ and fill this field with proper point in the repository.

IMPORTANT! Changes are not saved until blue save button on changed row is clicked, you can see enabled button when there are unsaved changes.

### Release Decision

The Release Decision can be either a ‘GO’ or ‘NO GO’. Currently the only rule has to be satisfied to get a ‘GO’ decision is that the release has to be approved.

## Release Execution

After the release is planned, approved and of course at the time of release, you can switch to release execution tab, which is on the same release page, right below release labels with basic information. On this page you will find all necessary tools to complete the release, gather all the release metrics, as well as sign off and close out the release. You can also see deploy measurement if deployment tool plugin provides such information.

### Release Tasks

You can see same release tasks list in simpler version, you can’t edit task or change order of tasks. The only one operation available is to complete task, by clicking the complete button next to release task. On small resolution screens complete button might be hidden, you just need to scroll right to see it.

### Release Process (metrics)

A series of buttons that are supposed to be clicked as a phase of the release is completed. This list may vary depending on release type and if the release requires the site to be put into maintenance mode (look at scheduling release). The metrics gathered from the process of marking off the completed phases of the release are manual, ReMi is not integrated with any external tool to put the site into maintenance mode or start deploys jobs.

### Signing off and closing release

On the release execution page you can sign off the release. If you are one of the users on sign offs list, you should be able to click ‘Sign off’ button next to your name. You don’t need to be logged in to sign off release.

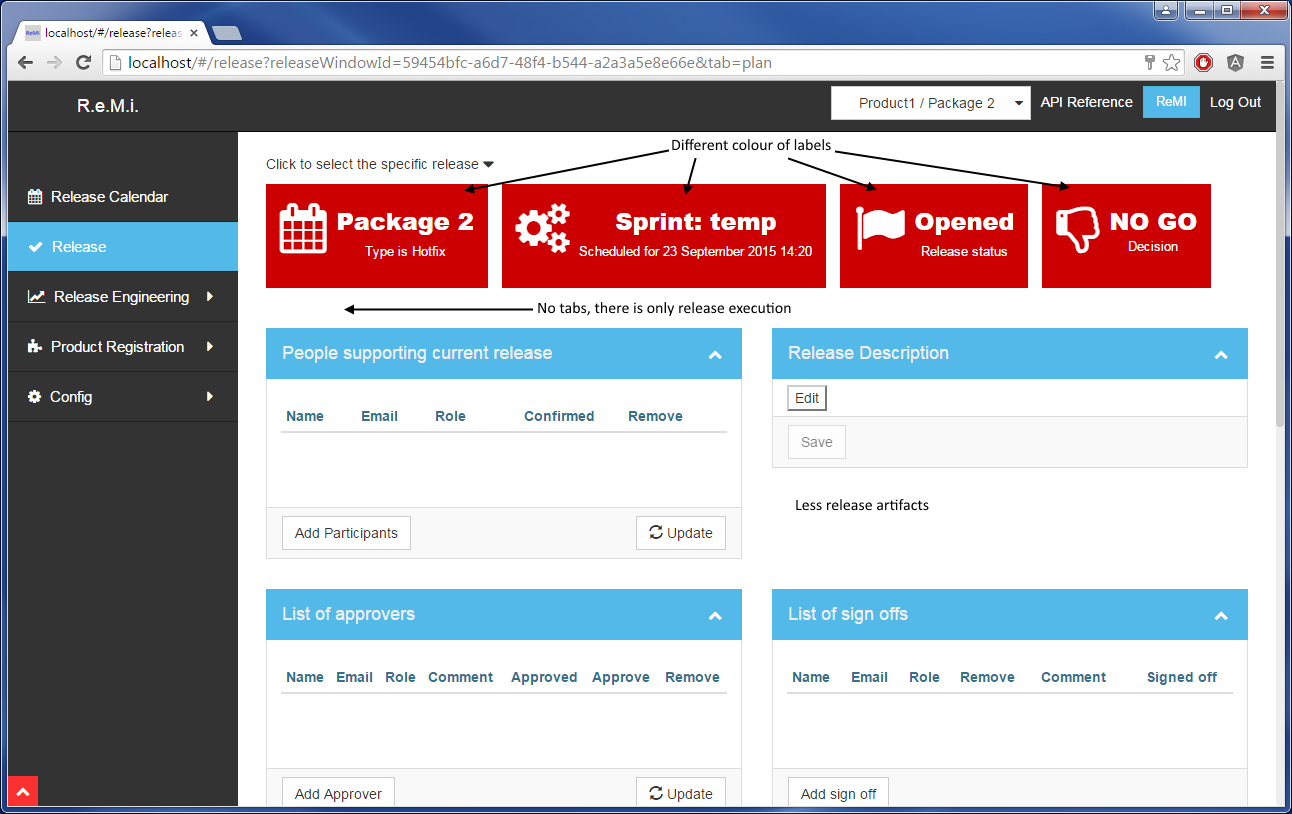
### Release Jobs Measurements

After the release is closed and provided that the Deployment Tool plugin is configured for current package, the Release Jobs Measurements will be imported to ReMi. This table contains all necessary metrics about the deploy jobs. If a deploy job has more hierarchical structure it will be shown in the table tree.

## Other release types

For different release types the Release page layout may be different. Below the main differences are described.

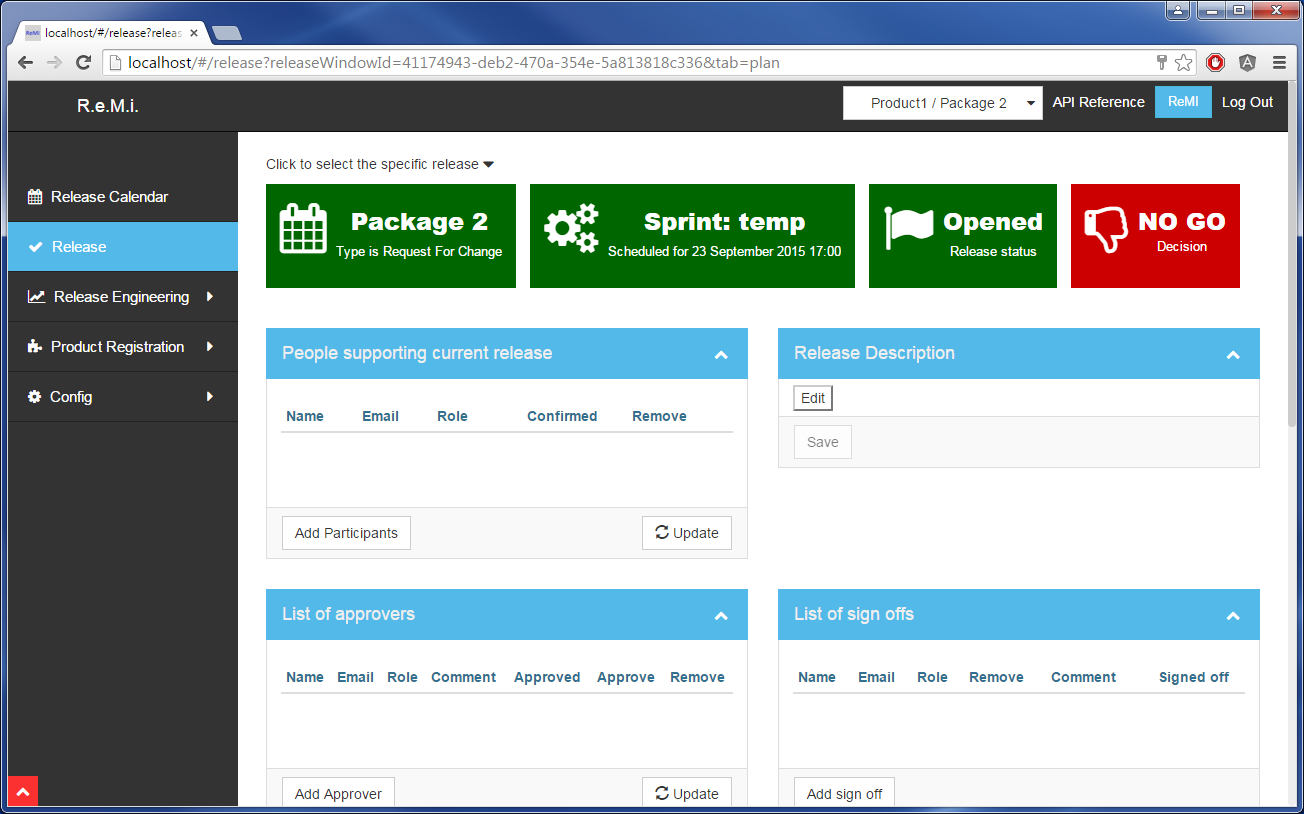
### Hotfix



The screenshot of the HotFix page above has the following differences to a standard release:

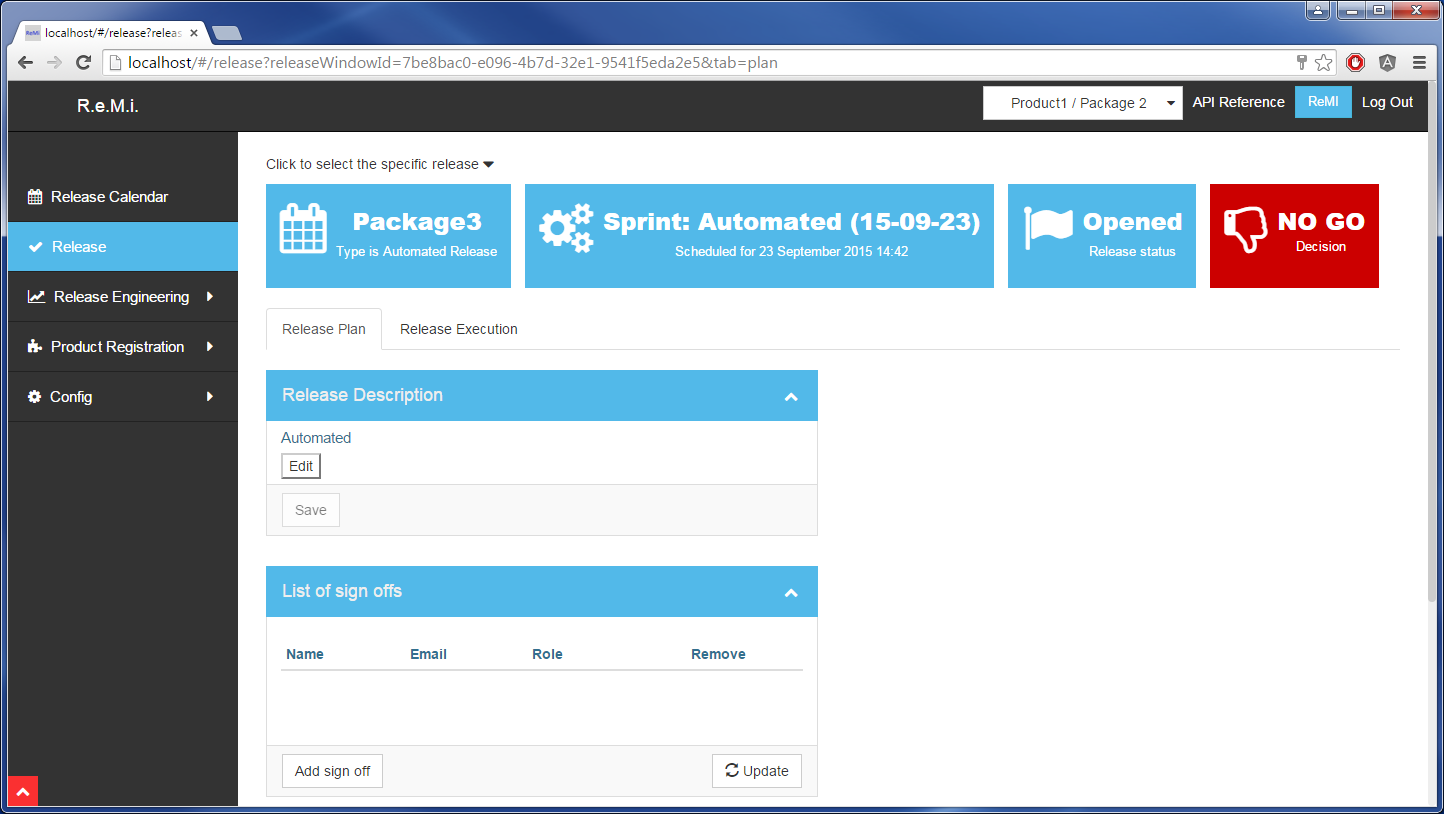
* Labels are coloured red
* There is no plan and execution tab as a hotfix is conducted urgently
* There is no Release Content, Release Jobs or Release Changes.

### Request for Change



Request for Change has the same layout as hotfix and the labels are coloured green.

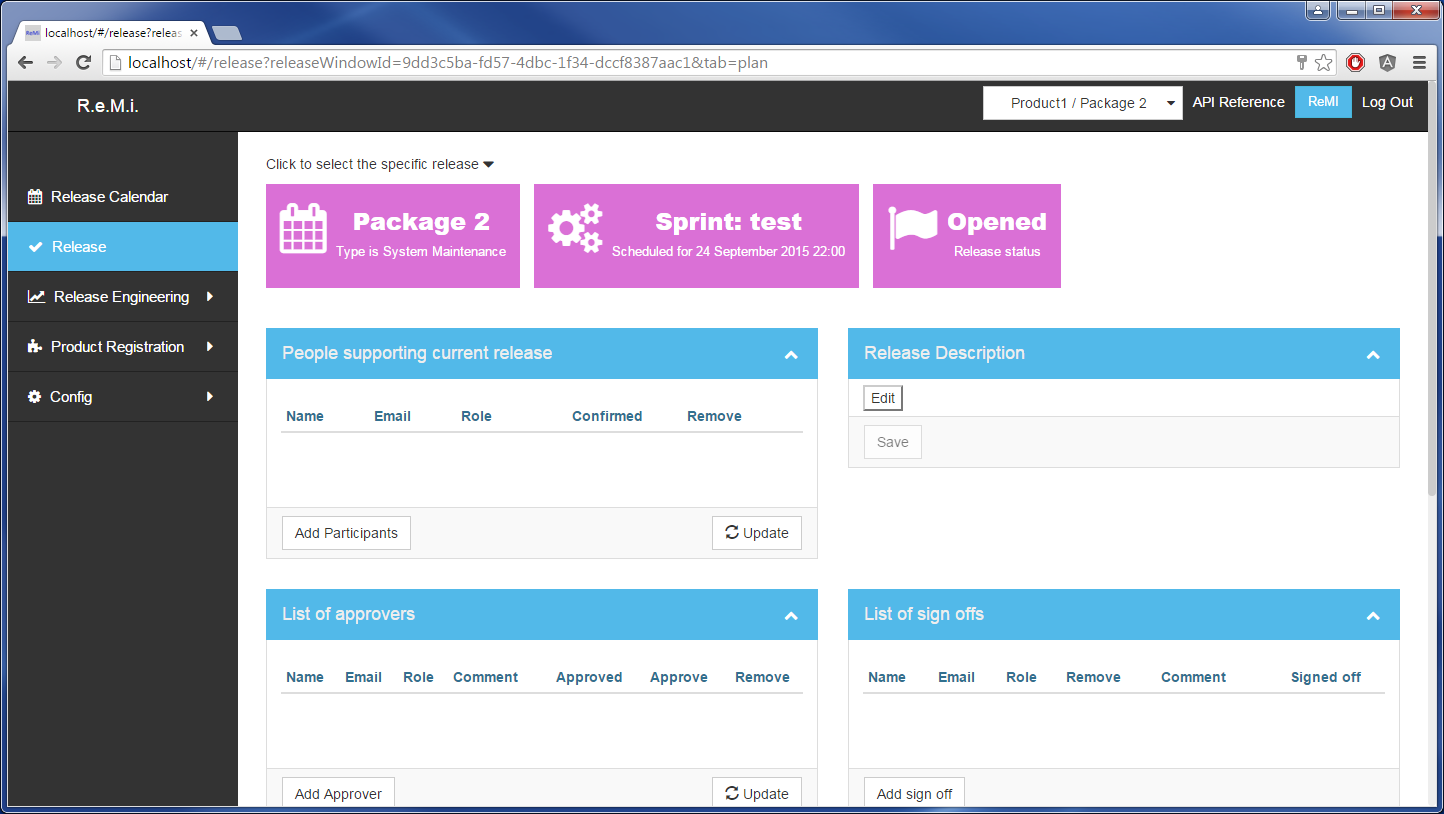
### Automated Release



Automated releases can be only created through the ReMi API, if package is on Automated Release Track (Look at Business Units configuration). Automated release is simpler version of Scheduled Release, the differences are:

* Colour of label is light blue.
* There are also two tabs for planning and executing release
* There are no Release Participants.
* There are no releases approvers – automated releases are by default pre-approved.
* There are no release tasks – automated releases do not have manual tasks.
* There is no Release Content, Release Jobs or Release Changes.

### System Maintenance, PCI, 3rd Party and Corp IT



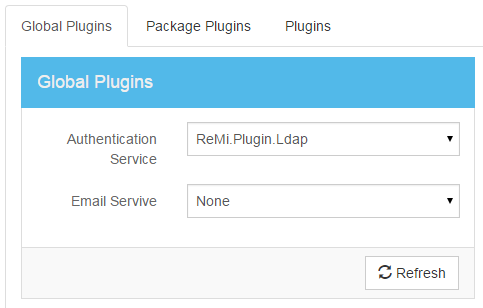
All 4 types of System Maintenance have the same layout. The colour of labels might be slightly different. The main difference from Scheduled Release:

* Colour of labels
* No release decision
* No tabs for planning and executing
* There is no Release Content, Release Jobs or Release Changes.

# Basic plugins configuration

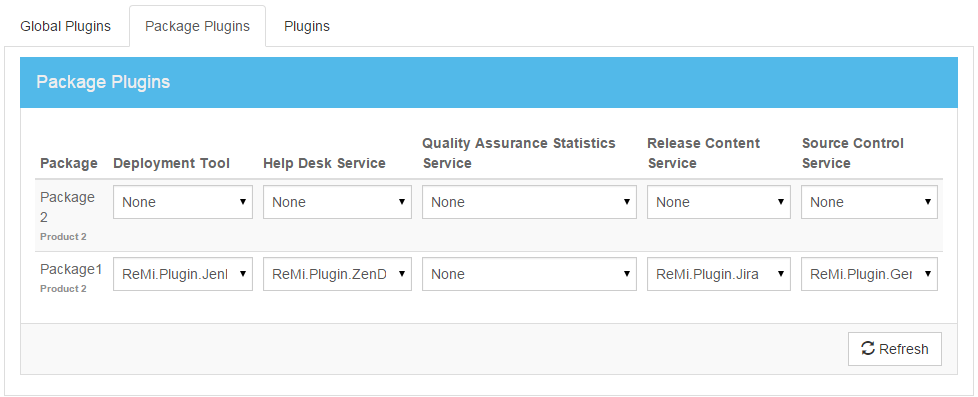
## Global plugins configuration

Global plugins are the plugins that are used across whole application, regardless of package or any other factors. Currently ReMi works only with two global plugins which is authentication and e-mail plugin. As you can see in the screenshot below, you can choose one plugin from the select box or none.



## Package plugins configuration

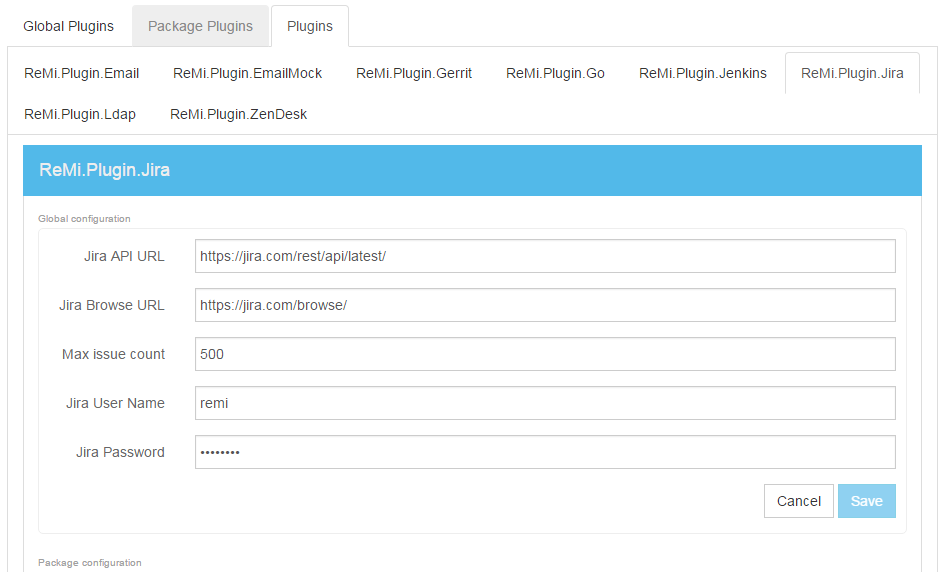
Package plugins are configured per package. So for example a deployment tool, such as Jenkins or GO, can be used by according to the package preferences (or rather team that is conducting releases). Below in the screen capture you can see the package plugins configuration. Each package can have a different set of plugins associated with it.



## Jira plugin

The Jira plugin is a Release Content Service plugin. It is used import all the features which are going to be released. Each package plugin can have global configuration, for example credentials for the Jira API. The Jira plugin can be configured as follows:

* Jira API URL – host address to Jira RESTFul API
* Jira Browse URL – host address base for tickets. It helps to construct a link directly to the ticket, which in this case will be for example: https://jira.com/browse/RM-123
* Max issues count – if the JQL filter is not well tailored for its needs it might pick up too many tickets. To prevent this, you can restrict amount of tickets which will be retrieved from API.
* Jira User Name and Password – Credentials for user which will be able to use Jira API.



Each package plugin also has separate configuration specific to each package, for example Jira tickets JQL filter. With the Jira plugin you can configure:

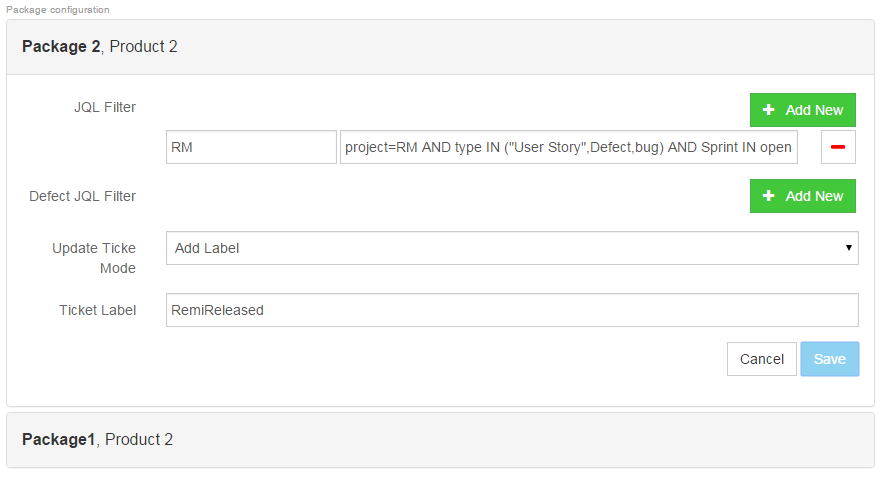
* JQL Filter – user can add as many JQL filters as needed, but it’s not recommended to create too many. Good practice is to keep one filter per package (team) and release single piece of software at once. To find out how to create JQL filter please see the advanced searching Jira documentation. As an example the following constitutes a valid JQL query:

*project=RM AND type IN ("User Story",Defect,bug) AND Sprint IN openSprints() AND labels != "RemiReleased"*.

That filter takes all tickets from project RM, which is one of type: User Story, Defect or Bug. It’s in open spring and it does not have ‘RemiReleased’ label.

IMPORTANT! Make sure different packages are not getting the same tickets, this may cause unexpected results.

* Defect JQL filter – This feature is currently not in use.
* Update Ticket Mode – user can specify action, which will be triggered on each ticket included into release, after release is closed. Now there is only one action, which is adding label to the ticket (or of course none action).
* Ticket Label – if ‘Add Label’ action is selected, this is text of the label which will be added to the ticket.

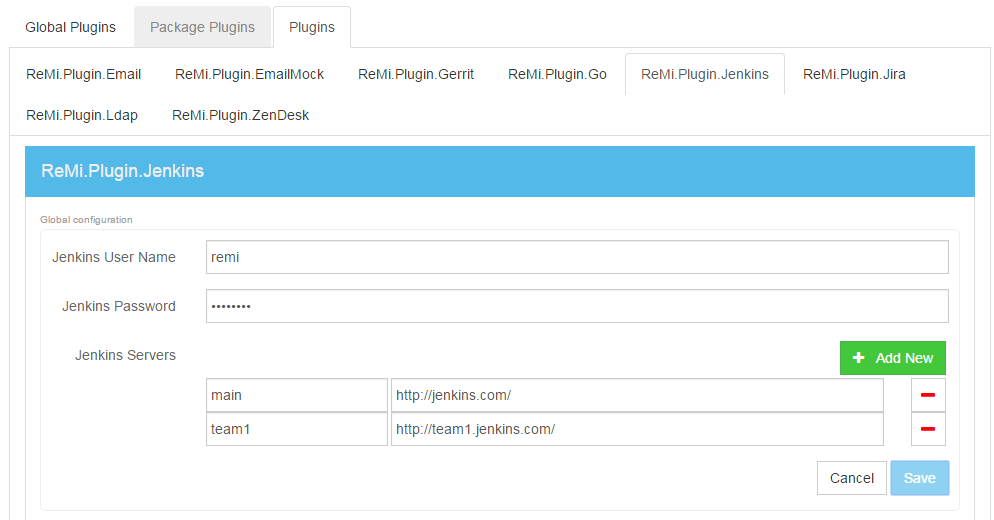


## Jenkins plugin

Jenkins is continuous integration tool, which can also be used to trigger deploys to a production environment. Configuration for this plugin also contains two sections, global and packages.

Global configuration:

* Jenkins User Name and Password – credentials required to call Jenkins RESTful API.
* Jenkins Servers – user can define here list of servers used for packages. One server can be used by 0 or more packages. Each server has name and URL address to Jenkins server.



Package configuration:

* Jenkins sever – select one from the list defined in the global configuration. If recently added server is not showing up on select list, please refresh page.
* Jenkins Jobs – this is list of jobs defined in Jenkins server for production deployment. The configuration is in JSON format.

Example:   
**[**

**{**

**"Name":** "ReMi\_Live-deploy",

**"IsIncludedByDefault":** true,

**"IsDisabled":** false,

**"ExternalId":** "d66288c0-d07d-45f4-9eec-69226b5ff6ee"

**},**

**{**

**"Name":** "ReMi\_Live\_FE-deploy",

**"IsIncludedByDefault":** true,

**"IsDisabled":** false,

**"ExternalId":** "00eb30a0-6079-11e5-9d70-feff819cdc9f"

**}**

**]**

* + Name – name of Jenkins job,
  + IsIncludedByDefault: if true, job will be included to each release by default.
  + IsDisabled – if true, job won’t be shown on new releases, it’s better to disable job than remove, because history remain.
  + ExternalId – unique UUID for each job. There are plenty online UUID generators, which can be used to generate it.
* Time zone – time zone of the Jenkins sever. This information can be found in Jenkins under System Information, time zone. It is recommended to use UTC (GMT) for Jenkins server configuration.

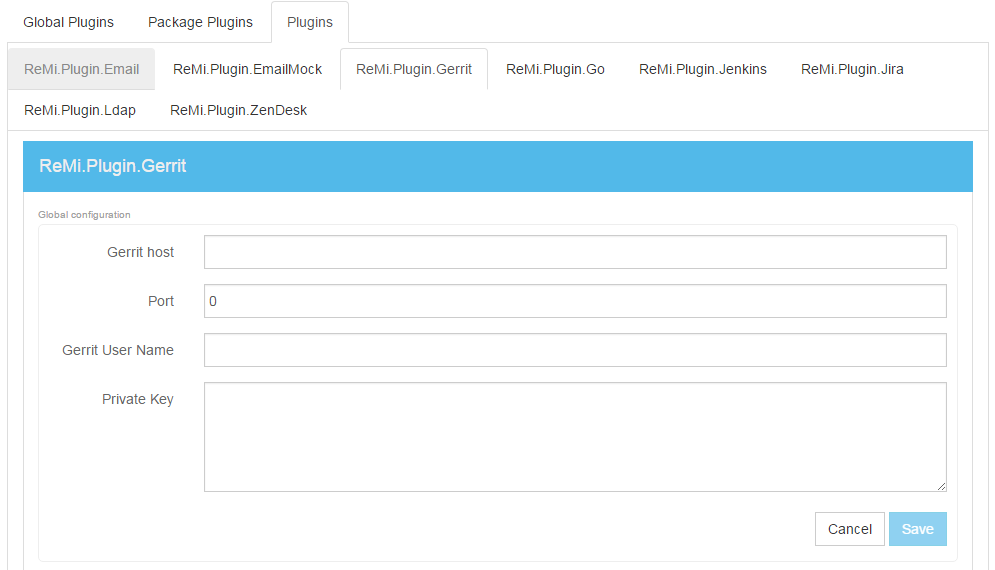
## GO plugin

GO plugin configuration looks similar to the Jenkins plugin configuration, the only difference is that GO package configuration does not have a time zone setting and the naming convention in GO is bit different, instead of jobs there are pipelines, but from a configuration view nothing is different.

The GO plugin has another feature; it may also have function as a Source Control Plugin that retrieves physical changes from the source code. The GO plugin retrieves code changes from down streaming pipelines.

## Gerrit plugin

The Gerrit plugin is another Source Control plugin for tracking source code changes per release. It is able to import git commits from the Gerrit API using the format *‘git log from..to’*, where the ‘from’ and ‘to’ parameters can be a commit hash, branch name, tag name etc. The Gerrit plugin also has global configuration for the API URL address, port and credentials( user name and private key).



For each package the following settings need to be configured:

* + ExternalId – unique repository id
  + Name – name of repository
  + StartFromLatest – if checked, changes by default will be taken from latest commit from previous release. This can be change for each release, on Release Plan page.
  + DefaultFrom – this is default point from where commits will be taken. This can be change for each release, on Release Plan page.
  + DefaultTo – this is default point to where commits will be taken. This can be change for each release, on Release Plan page.
  + IsIncludedByDefault – this specify if repository will be included by default to release. This can be change for each release, on Release Plan page.
  + IsDisabled – instead of removing repository, user can mark it as disabled; it won’t appear on new releases.

Sample JSON:

**[**

**{**

**"ExternalId":** "68af86da-befc-4623-affd-b396a9c3b75f",

**"Name":** "main/backend",

**"StartFromLatest":** true,

**"DefaultFrom":** "08d24285b02b560864cb88dd3ba51b78e477a41d",

**"DefaultTo":** "release",

**"IsIncludedByDefault":** true,

**"IsDisabled":** false

**},**

**{**

**"ExternalId":** "3e3accf9-44d4-4cef-b698-0d32c9b7e9b8",

**"Name":** "main/frontend",

**"StartFromLatest":** true,

**"DefaultFrom":** "6d14a9383772c93b81b078e70743035a45c91b08",

**"DefaultTo":** "release",

**"IsIncludedByDefault":** true,

**"IsDisabled":** false

**}**

**]**

## ZenDesk plugin

The ZenDesk plugin is used for integration with the HelpDesk system. This kind of tool can be used to add more information and data to release tasks. ZenDesk for example allows adding attachment to the ticket, which can be very helpful.

The ZenDesk plugin has only global configuration for the URL to its API and uses standard credentials for authentication.