Redmine ticket provider

Version 1.0.0

User's Guide

July 2, 2019

Remark

Revision history

Version	Date	Content
1.0.0	2019-07-02	First edition

Table of contents

1. **Introduction**

A Redmine ticket provider is a ORiN2 CAO provider that uses a REST to communicate with the Redmine to obtain and post Redmine tickets.

Redmine tickets can be viewed and posted using Redmine ticket providers.

This document outlines the Redmine ticket providers and describes the CAO interfaces (functional specifications) that are implemented in the ticket providers.

2. **Provider overview**

2.1. Operating system installation

The Redmine Ticket Provider module consists of the following DLLs. If you install it using the ORiN2 SDK installer, you do not need to install it. If you are installing manually, perform the procedure as shown in Table 2-1.

RegistAsm.bat and UnregistAsm.bat are located in the DotNet ¥BAT folder below the ORiN2SDK installation folder.

Tabular 2-1 Issues providers21

File name	CaoProvRedmineIssues.dll
ProgID	CaoProv.Redmine.Issues
Registry registration	RegistAsm.bat CaoProvRedmineIssues.dll
Unregistering the registry	UnregistAsm.bat CaoProvRedmineIssues.dll

2.2. Overview

Redmine tickets can be viewed and posted using Redmine ticket providers.

2.3. Method Properties

2.3.1. CaoWorkspace::AddController method

Create Controller classes and add them to the workspace.

Format AddController (<bstrCtrlName:BSTR>, <bstrProvName:BSTR>, <bstrPcName:BSTR>, [<bstrOption:BSTR>])

< bstrCtrlName >	:	[in] controller name
< bstrProvName >	:	[in] Provider name. Fixed = "CaoProv.Redmine.Issues"
<bstrPcName>	:	Running machine name of the in provider (unused)
<bstrOption>	:	In option string

You can specify the following options, separated by commas:

URL	:	Specifies the URL to connect to. Required Examples URL=http://loalhost/redmine/
UserID	:	Specifies the userid to log in to the Redmine. If specified, it must be set to Password.
Password	:	Specifies the passwords for users who log in to the Redmine. If specified, it must be set to UserID. Examples UserID=Test,Password=Test
AppKey	:	Specifies the keys for accessing the REST API of the Redmine. If you specify AppKey, then UserID and Password are ignored.

2.3.2. CaoController::AddExtension() commands

Create a Extension class and add it to the Controller class.

Extension classes correspond to Redmine tickets.

Format AddExtension(<VariableName>,<Option>)

<VariableName> : [in] variable name (VT_BSTR)

<Option> : In option string

Specify one of the following options:

ProjectID	:	Creates a new Extension by specifying the project to which you want to submit the new project.
IssueID	:	Creates a Extension in edit-mode by specifying the IDs of the tickets to be retrieved.

Returned value : [out] Added Extension

Example

```
CaoExtension ext = m_caoController.AddExtension("Ext", "ProjectID=207")
```

2.3.3. CaoExtension::AddVariable() commands

Create a Variable class and add it to the Extension class.
Variable classes correspond to items in Redmine tickets.

Format AddVariable(<VariableName>,<Option>)

<VariableName> : [in] variable name (VT_BSTR)
<Option> : In option string
For non-system variables, specify the following options:
CustomFieldID : Specifies the ID of the custom field that the variable represents.
Required for user-defined variables.
Returned value : [out] Added Variable

Example

```
CaoVariable ext = m_caoExtension.AddVariable("@ID", null)
```

2.3.4. CaoController::GetVariableNames() commands

Get a list of available system variables.

Format GetVariableNames(<Option>)

<Option> : In option (not used)
Returned value : [out] List of available system variables (VT_ARRAY | VT_BSTR)

Example

```
String[] varNames = _controller.GetVariableNames(null)
```

2.4. Variable classes

2.4.1. CaoVariable::get_Value()

Gets the value of a variable.

Format GetValue()

Returned value : Value of the [out] variable (VT_VARIANT)

Example

```
Object variableValue = val.Value
```

2.4.2. CaoVariable::put_Value()

Sets the value of a variable.

Format PutValue(<VariableValue>)

<VariableValue> : Value of the [in] variable

Example

```
Val.Value = "1"
```

3. Command reference

3.1. Controller classes

List of 3-1 CaoController::Execute Commands31

3.1.1. CaoController::Execute("GetProjects ")

Get a list of projects that the logged-in user can access.

Format GetProject()

Returned value : [out] Project information (VT_VARIANT | VT_ARRAY)
Each element of the array is VT_BSTR | VT_ARRAY, the first element is the project ID, and the second element is the project name.

Example

```
m_caoController.Execute("GetProjects", null)
```

3.1.2. CaoController::Execute("GetTrackers ")

Get a list of trackers.

Format GetTrackers ()

Returned value : [out] tracker information (VT_VARIANT | VT_ARRAY)
Each element of the array is VT_BSTR | VT_ARRAY, with the first element being the tracker ID and the second element being the tracker name.

Example

```
m_caoController.Execute("GetTrackers", null)
```

3.1.3. CaoController::Execute("GetStauses ")

Gets a list of statuses.

Format GetStatuses()

Returned value : [out] status information (VT_VARIANT | VT_ARRAY)
Each element of the array is VT_BSTR | VT_ARRAY, with the first element being the status ID and the second element being the status name.

Example

```
m_caoController.Execute("GetStatuses", null)
```

3.1.1. CaoController::Execute("GetCategories ")

Gets a list of the categories set for the specified project.

Format GetCategories(<projectId>)

<projectId>	:	ID of the project for which you want to get the in category
Returned value	:	[out] category information (VT_VARIANT | VT_ARRAY) Each element of the array is VT_BSTR | VT_ARRAY, and the first element is a category ID and the second element is a category name.

Example

```
m_caoController.Execute("GetCategories", 207)
```

3.2. Extension classes

List of 3-2 CaoExtension::Execute Commands3

3.2.1. CaoExtension::Execute("Apply ")

Posts to the Redmine using the values specified in the Variable.

When submitted in New mode, the variable contents are updated with the information of the submitted ticket, and the mode automatically changes to Edit mode.

Format Apply(<Comment>)

<Comment> : [in] The comment to be posted. It is ignored in the newly created mode.

Returned value : No [out]

Example

```
_extension.Execute("Apply", "Comments")
```

3.2.2. CaoExtension::Execute("CreateNew ")

Initializes the Extension in Create New mode, specifying the projects to post, and returns to the same state as when you specified ProjectID in AddExtension.
The added variables are not deleted and only their values are restored to their initial values.

Format CreateNew(<ProjectID>)

<ProjectID> : ID of the project to which you want to submit [in]
Returned value : No [out]

Example

```
_extension.Execute("CreateNew", 207)
```

3.2.3. CaoExtension::Execute("Reload ")

Initializes the Extension in edit mode by specifying the ticket to be edited, and returns the ticket to the same state as when IssueID is specified in AddExtension.
The added variables are not deleted and only their values are restored to their initial values.

Format Reload(<IssueID>)

<IssueID> : ID of the ticket for which you want to edit [in]
Returned value : No [out]

Example

```
_extension.Execute("Reload", 1000)
```

3.2.4. CaoExtension::Execute("GetCustomFields ")

Get a list of custom fields.

Values can be retrieved only in Edit mode (with ticket information loaded).

Format GetCustomFields(<Option>)

<Option>	:	In option (not used)
Returned value	:	Returns information about a custom field in [out] (VT_VARIANT | VT_ARRAY). Each element is a three-element array (VT_VARIANT | VT_ARRAY) with the following meanings: First element: Custom field ID (VT_I4) Second element: Custom field name (VT_BSTR) Third element: Custom field value (VT_BSTR)

Example

```
Object customFields = _extension.Execute("GetCustomFields", null)
```

3.2.5. CaoExtension::Execute("RegistAttachmentFile ")

Registers the file to be attached to the ticket.

Saved files are uploaded when you execute Apply and attached to tickets.

Format RegistAttachmentFile(<Option>)

<Option> : Pass the following data in [in] (VT_VARIANT | VT_ARRAY).

First element: File image (VT_UI1|VT_ARRAY)
Second element: File name (VT_BSTR)
Third element: File description (VT_BSTR, optional)
Fourth element: Content type (VT_BSTR, optional)

If the content type is omitted, the content type is automatically determined according to the extension of the specified file name.

Returned value : No [out]

Example

```
_extension.Execute("RegistAttachmentFile", new object[] {data, name})
```

4. About Variables

4.1. Overview

There are two types of Redmine ticket-provider variables: system variables, which correspond to items that are standard in the Redmine, and user-defined variables, which correspond to custom fields.
Providers keep track of the values set in their variables put_Value, and only the recorded values are sent to the Redmine when Apply commands are executed.
After the Apply command is executed, the recorded value is deleted, and the previous value is not sent the next time the Apply command is executed.

4.2. System variable

The following list lists the system variables that you can use with Redmine ticket providers.
System variables include variables that set posted values that can be written to, and variables that are dedicated to references that cannot be written to.
If the writable item is omitted, the default value is used when the default value is set in the new creation mode, and the value is not changed in the editing mode.
In addition, the name of the item in which the ID and the name are paired can be obtained only when the value is not set to the ID in the edit mode and the value is not set to the ID in the "put_Value", otherwise the value becomes empty.

TABLE 4-1 List of System Variables1

System variable	Meaning	Writing	Remark
@ID	Ticket ID	Not possible	
@Subject	Title	Possible	Required
@ProjectID	Project ID	Possible	
@ProjectName	Project name	Not possible	
@TrackerID	Tracker ID	Possible	When VT_NULL is set during editing, the values do not change during Apply.
@TrackerName	Tracker name	Not possible	
@StatusID	Status ID	Possible	When VT_NULL is set during editing, the values do not

			change during Apply.
@StatusName	Status name	Not possible	
@PriorityID	Priority ID	Possible	If you set a priority ID that does not exist at the time of editing, an error occurs at the time of Apply.
@PriorityName	Priority name	Not possible	
@AuthorID	Employee submitting message ID	Not possible	
@AuthorName	Name of the contributor	Not possible	
@AssignedToID	Person-in-charge ID	Possible	When VT_NULL is set during editing, the values do not change during Apply.
@AssignedToName	Name of the person in charge	Not possible	
@CategoryID	Category ID	Possible	
@CategoryName	Category name	Not possible	
@ParentIssueID	Parent ticket ID	Possible	If VT_NULL is set during editing, the VT_NULL is deleted when the Apply is executed.
@Description	Description	Possible	
@StartDate	Start day	Possible	If VT_NULL is set during editing, the VT_NULL is deleted when the Apply is executed.
@DueDate	Date	Possible	If VT_NULL is set during editing, the VT_NULL is deleted when the Apply is executed.
@DoneRatio	Progress	Possible	Integer in percent Set to 0 when Apply is executed when VT_NULL is set during editing.
@EstimatedHours	Planned man-hours	Possible	Real If VT_NULL is set during editing, the VT_NULL is deleted when the Apply is executed.
@CreatedOn	Creation date and time	Not possible	
@UpdatedOn	Updated date	Not possible	
@IsNewIssue	New creation mode flag	Not	True in New mode or False in Editing mode

possible

4.3. User-defined variables

User-defined variables correspond to custom fields set in a ticket.

All user-defined variables are writable.

When adding user-defined variables, be sure to specify "CustomFieldID={custom field identifier}" in the options string.

User-defined variables with duplicate custom field IDs cannot be added.

5. **Sample program**

Here are a few simple examples of programs using Redmine ticketing providers and C#.

Sample program

```
Try
{
    _ctrl = _engine.Workspaces[0].AddController(
        "Redmine",
        "CaoProv.Redmine.Issues",
        Null,
        String.Format(      "Url={0},
                            UserID={1},Password={2}",
                            TxtUrl.Text,
                            TxtUserId.Text,
                            TxtPassword.Text));

    _extension = _ctrl.AddExtension(   "Issue",
                                                String.Format("ProjectId={0}",
                                                TxtProject.Text));
    _extension.Variables.Add("@ID", null);

    Foreach (Control ctrl in pnlEdit.Controls)
    {
        RedmineEditControl editCtrl = ctrl as RedmineEditControl;
        If (editCtrl == null)
        {
            Continue;
        }

        _extension.Variables.Add(editCtrl.VariableName, editCtrl.Option);
    }

    CreateNew();

    TxtProject.Enabled = false;
```

```
        TxtPassword.Enabled = false;
        TxtUserId.Enabled = false;
        TxtUrl.Enabled = false;
        BtnConnect.Enabled = false;
        PnlEdit.Visible = true;
    }
    Catch (Exception ex)
    {
        If (_ctrl != null)
        {
            _engine.Workspaces[0].Controllers.Remove(_ctrl.Name);
        }
        MessageBox.Show(    This,
                            Ex.ToString(),
                            This.Text,
                            MessageBoxButtons.OK,
                            MessageBoxIcon.Error);
    }
```