2014

Templar Widget Management Service Developer Guide

The purpose of this document is to:

- Understand how to use Templar widget management service.
- This document is intended for Developers.

Sushant Salunkhe
Tavisca Solutions Pvt. Ltd
1/10/2014

If you have comments /queries about this documentation, email them to:

templar@tavisca.com

Table of Contents

Introduction

As the name suggests Templar Widget Management Service (TWMS) is used to programmatically manage all widget related activities for Templar deployment.

At a high level it provides features like the ability to get and manage widgets on a deployment.

How to access it?

The TWMS can be accessed at /Templar/Services/TemplarWidgetManagementService.svc url on the base Templar deployment.

To use TWMS, you will need to create a client and use it to call the TWMS. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe
http://templarDeployment.com/Templar/Services/TemplarWidgetManagementService.svc?wsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

```
class Sample
{
    static void Main(string[] args)
    {
        TemplarWidgetManagementServiceClient client = new TemplarWidgetManagementServiceClient();

        // Use the client variable to call operation on the service.

        // Always close the client.
        client.Close();
    }
}
```

Recommendations

1. Always host it on SSL as clear-text password is sent as part of request.
2. Avoid using it on production environment as some methods may be time consuming.

Common Parameters

Common Request Parameters

Every request to the service must contain authentication information described below.

Name	Type	Description
Username	String	Templar username for the deployment
Password	String	Password for the above specified username

Common Response Parameters

Every response from the service contains following information related to the status of operation performed.

Name	Type	Description
IsSuccessful	Bool	Is true if operation was performed successfully, else false.
Message	String	Contains message corresponding to status of the operation performed.
ResponseCode	Int	Response code corresponding to status of the service call.

Common Error Messages

Code	Message	Description
401	Authentication failed	Server was not able to authenticate using username and password provided.
500	Error occurred while processing your request	Some unexpected error occurred while performing the operation on server. Check server exception logs for further details.

Service Methods

1. GetWidgetList

Returns list containing metadata of widgets at current server.

Request Object Structure: GetWidgetListRQ
AuthenticationInfo and EnvironmentInfo (optional) are required parameters to call this method. Sample request below describes the request object formation.

Response Object Structure: GetWidgetListRS
Along with common response parameters, following information is also returned as a part of response object.

Name	Type	Description
Widgets	List<Widget>	List containing metadata of the widgets.

Sample Request Object

```
var request = new Proxy.GetWidgetListRQ()
{
    AuthenticationInfo = new Proxy.AuthenticationInfo
    {
        Username = "username",
        Password = "password"
    },
    // server information is optional to provide to get
    // metadata from current deployment
    Server = new Proxy.EnvironmentInfo
    {
        Location = "server location",
        UserName = "server credential",
        Password = "server credential"
    }
};
```

Sample Response Object

```
<Widgets>
    <Widget>
        <Id>11018</Id>
        <Name>AsyncSampleWidget</Name>
        <Description>Sample Widget Async Demonstration</Description>
        <Icon>~/Widgets/</Icon>
        <State>&lt;state&gt;&lt;/state&gt;</State>
        <Url>~/Widgets/Demo/AsyncSample/AsyncSampleWidget.ascx</Url>
        <Cultures />
    </Widget>
    <Widget>
        <Id>11019</Id>
        <Name>TravelNewsWidget</Name>
        <Description>Allows side by side widget placement</Description>
        <Icon>~/Widgets/News/news.png</Icon>
        <State></State>
        <Url>~/Widgets/News/TravelNewsWidget.ascx</Url>
        <Cultures />
    </Widget>
<Widgets>
```

Detailed Object Model

Widget

Name	Type	Description
Id	Long	Id of the widget
Name	String	Name of the widget
State	String	Default state of the widget
Url	String	Url of the location of widget
Icon	String	Url for icon file of the widget
Description	String	Description of the widget
Cultures	List<Culture>	List of cultures of the widget