User Services

Introduction

*f*Series provides web service methods for each of the main roles (user, designer, data, admin). These may be used directly, for example to generate documents from within code. The User Service class is helper code that handles the interaction with user role web services. This document describes the methods available and how to use them.

Connection and Authentication

User Service lets you connect to the web service and maintains the connection throughout the session.

Connecting to the Server

To make a connection to the server use the following method:

```
UserServices.ConnectToServer("[Domain]", "");
```

The first parameter is the URL of your fSeries server. The second parameter (Organisation) is not currently in use so should be set as shown.

To check the connection at any time use:

```
UserServices.Connected
```

This returns true/false according to whether you have a current connection.

Automatic Authentication

When you connect to server User Services will attempt to auto-authenticate the user if either:

a) You have already supplied UserId and Password values (see properties), or

b) The server setting "AutoAuthenticate" set to true (using the logged in user's identity)

Use the "Authenticated" property to test if the user is currently authenticated.

Authentication

Four values are sent to the server for authentication:

- User Id
- Password (which is automatically encrypted)
- Auth Factors
- Security Group

User Id and Password

You can either set these with UserId and Password properties prior to calling Authenticate() method, or supply as parameters to Authenticate(UserId, Password).

Note that if you need to authenticate based on the logged in user's identity you cannot use this method; it must be done at connection via auto-authentication.

Auth Factors

This is a collection of values that are retained throughout the session and may be accessed by fData when gathering data.

By default they include the following:

- Organisation (not currently used)
- Instance (see properties)
- Role ("User")
- Context Data (see properties)

In addition you can add a collection of other values by adding key/values to the UserDefinedAuthFactors hash table;

```
UserServices.UserDefinedAuthFactors.Add("MSCRMUser", "12fb6a43-4473-e011-8720-00155da5304e");
```

In this example the MSCRM caller id is being set in order for the data gathering plugin to act on this user's behalf. By setting the key "MSCRMUser" to a user's GUID the plugin can access this to act accordingly.

You can add any values in the set in order to communicate them to data gathering processes.

Security Group

This is a feature of fSeries that enables different methods of authentication within the same implementation. If Security Groups are in use, set the SecurityGroup property prior to Connection/Authentication.

Maintaining the Session

The Authenticate web service method returns an Auth Token string that contains all session data including the current time to detect timeout (set in minutes as the AuthTokenTimeout setting).

The timeout is reset with every server interaction but if a lengthy period of inaction is likely simply call the Authenticated property as this renews the Auth Token

Note that this will not re-authenticate, only extend an existing valid Auth Token.

Document Generation

The most likely use of User Services is to generate documents from within code, for example as part of a workflow. Two methods are provided to generate a specified document; one uses a comma separated list of key/values as parameters, the other uses an XMl document to pass parameter values;

public byte[] Generate(string Path, string FileName, string Parameters, string Format)
public byte[] Generate(string Path, string FileName, int Mode, XmlNode Parameters, Nullable<DateTime> Effective, string Format)

Path is the optional path to the document template. This may be specified within the FileName parameter.

FileName is the name and (optionally) full or partial path to the document template.

Path and FileName must between them locate the document template within the Published folder, unless the Mode is specified other than 0 (0=Published [default], 1=Test folder).

Format is the required format of the generated document (e.g. docx, pdf). The list of available formats can be obtained from the AvailableFormats() method which returns an Xml Node with the list of available format ids and names.

String Parameters

This method is the simplest to use for straightforward generation. The Parameters argument contains a comma separated list of parameters required by the data gatherer. For example

```
“quoteid=12fb6a43-4473-e011-8720-00155da5304e”.
```

One special parameter is “effective” which is a date and sets the fData Effective property. If the DSD that gathers the data for the document requires an effective date, pass it using this parameter name.

Xml Node Parameters

The more detailed Generate method takes two arguments for parameters, one an Xml Node with parameter value, the other a nullable data for the fData Effective date property.

The format of the parameters Xml Node is:

```
<entries>
            <entry id=”quoteid” value=”12fb6a43-4473-
e011-8720-00155da5304e”/>
</entries>
```

Any number of parameters may be passed this way but the ids must match exactly the required user entries of the DSD that gathers data for the requested document template.

Compatibility

The User Service class has a hard-coded compatibility version that can be checked against the server to ensure that it is compatible with the server version.

```
UserServices.Compatible();
```

Returns true/false. Use the CompatibilityVersion property to see the classes version number.

Other Methods

A number of additional methods are provided.

```
public XmlNode AllData(string DSD, string Values)
```

Returns an Xml node gathered using the specified DSD with parameters supplied as a comma separated list of values.

```
public XmlNode Data(string DSD, DateTime Effective, string SchemaValues,
string UserEntries, out string OutputFile)
```

Returns an Xml node gathered using the specified DSD with details parameter. Also returns the file id that would be given to a generated document based on the parameters given.

```
public XmlNode GetData(string DSD, string DataGroup, string Values)
```

Returns an Xml Node gathered using the specified DSD/Data Group with parameters supplied as a comma separated list of values.

```
public XmlNode LookupList(string DSD, string DataGroup, string LookupValue,
string LookupText, string Values)
```

Returns an Xml Node with a list if lookup values (value/text) from the DSD/Data Group based on the parameter values supplied. Used to obtain user entry options.

```
public XmlNode SchemaValuesSpec(string DSD)
```

Returns an Xml Node with a list of the specification for user entries required by the DSD. Used in user interface projects to discover the required values to be requested.

```
public XmlNode UserEntriesSpec(string DSD)
```

Returns an Xml Node with a list of the specification for user entries required by the DSD. Used in user interface projects to discover the required values to be requested.

Settings and Properties

The following settings and properties are available (those related to login options are documented in the next section).

Authenticated	Is the user currently authenticated
AuthToken	The current auth token
CompatibilityVersion	The compatibility version number of this version of the class
Connected	Is the class currently connected
ContextData	Used by the fSeries Audit Log option to record the context of data gathering. Must be set to a non-blank value. Default is “User”
DefaultInstanceName	If multiple server configuration instances are in operation, this returns the name of the default instance.
DefaultInstanceNotes	If multiple server configuration instances are in operation, this returns the notes of the default instance.
Instances	Returns an Xml Node of available server configuration instance id, names and notes
IsAdmin	Does the current authenticated user have the role of Admin
OrganisationName	Not currently used
Password	Sets the password of the authenticated user. Required if auto-authentication is not possible. Will be encrypted prior to sending to the server.

SecurityGroup	Sets the security group under which authentication is to be carried out.
ServerCompatibilityVersion	The current compatibility version number of the server.
ServerName	The name of the connected fSeries server
Unencrypted	If set to true, disables automatic encryption of the password when sent to the server
UserDefinedAuthFactors	A hashtable to which user defined key/values may be added to pass to the data gatherer
UserId	Sets the user id of the authenticated user. Required if auto-authentication is not possible.

Login Options

If your project includes a user interface you may wish to offer login options, in which case you need to know which options are valid for the connected server..

What options are valid?

The following are used to discover which options may be offered.

```
public bool AllowChangeAdminPassword(string Instance)
public bool AllowChangeUserPassword(string Instance)
public bool AllowLogin(string Instance)
public bool AllowSendPassword(string Instance)
```

In each case the configuration instance must be provided (may be blank).

Send User Information

The following methods are used to action user requests.

```
public bool SendUserId(string Instance, string Email, out string Response)
```

Sends an email to the specified address containing the user id(s) held for the email address. Response returns any response from the email sender.

```
public bool SendPassword(string Instance, string UserId, out string Response)
```

Sends an email to the specified user with their password. Response returns any response from the email sender.

Sample Code

```
fSeries.UserService UserService = new fSeries.UserService();
```

```
UserService.UserDefinedAuthFactors.Add("MSCRMUser", "12fb6a43-4473-
e011-8720-00155da5304e");
UserService.UserId = "ProxyUserId";
UserService.Password = "ab+38szAc*";
UserService.ConnectToServer("http://MyDomain/fSeries", "");
byte[] doc = UserService.Generate("Quotes", "FullQuote.docx",
"quoteid=0e3f892e-9a35-e211-a74a- 00155d000e01", "pdf");
```

This example adds the MSCRMUser caller id user GUID to the auth factors in order for it to be used in the data gatherer, then connects to the server, authenticating with the proxy user's credentials. Finally a byte array is returned for the binary file of a document generated in PDF format for the quote specified.

If it is not possible to use the User Service dll the following sample code is the equivalent to the above using the web service directly.

```
string AuthFactors = "Organisation=&Instance=&Role=User&ContextData=User&MSCRMUser=12fb6a43-4473-
e011-8720-00155da5304e ";
Service oService = new Service();
oService.Url = "http://MyDomain/fSeries/WS/fSeries.asmx";
string AuthToken = oService.Authenticate(AuthFactors, "ProxyUserId", "ab+38szAc*", "");
XmlDocument Parms = new XmlDocument();
Parms.LoadXml("<?xml version='1.0'?><entries><entry id="quoteid" value="0e3f892e-9a35-e211-
a74a-00155d000e01" /></entries>");
if (oService.Connected("")) byte[] doc = oService.Generate(AuthFactors , AuthToken , "Quotes",
"FullQuote.docx", 0, Parms.DocumentElement, null, "pdf");
```

Note that the AuthFactors must be created manually and the AuthToken retained to maintain state from call to call, and the parameters must be formed into an Xml Node.

In this example the password is passed unencrypted. This assumes the setting WSUnencrypted is set to true. If not, encrypt the password as follows (if unknown then always encrypt as the server will attempt decryption in case):

```
ClientSecurity oSec = new ClientSecurity();
string AuthToken = oService.Authenticate(AuthFactors, "ProxyUserId",
oSec.Encrypt("ab+38szAc*", "[default]"), "");
```