

Server Utility Plug-in for Servoy – “mfServerUtils 1.0”



By [Arup Ranjan Sahoo](#)

Mindfire Solutions

Introduction

“mfServerUtils”, is a Servoy Plug-in that gives the power to developer to control the Application Server and any connected Clients, running on the server. The Plug-in is smart enough to give you the capability to control the client’s (Rich Client & Web Client) behavior such as alerting Client, shutting down Client etc. Not only about the Clients can you also control your Servoy Application Server such as alerting all Clients, shutting down the server, restarting the server etc. And all these powerful things right from your Servoy Solution it self. For the security reason, I have added one property called "com.mindfire.plugins.mfserverutils.accesscode" to the Servoy Properties. I recommend, this property should be set and checked while calling any methods of the Plug-in.

Some Silent Feature of the Plug-in...

- Getting all/specific Client connected to the Application Server
- Shutting down Application Server
- Restart Application Server
- Alerting all the clients connected to the application server
- Checking Client is alive or not
- Shutting down clients
- Alerting clients
- ...and much more to be added

Plug-in Details

Plug-in Name: mfServerUtils

Plug-in Version: 1.0 - [09-02-2008]

Tested in Servoy Version: 4.1.X

Tested in Java Version: 1.5. +, 1.6. +

Tested in Servoy Client: Rich Client, Web Client

Tested in Platform: Windows XP, Mac OS X 10.5.5, Ubuntu Linux 8.10.

Price

The Plug-in is free.

Benefits

- Can be used for controlling Client's behavior such as alerting Client, shutting down Client etc. straight with in your Servoy solution.
- Can be used to get the list of all currently connected Clients and other remote Clients.
- Can be used to communicate in between Clients by alerting Clients.
- Can be used for controlling Application Server's behavior such as alerting all Clients, shutting down/restarting Application Server etc. straight with in your Servoy solution.
- Much more functionality to be added to make this Plug-in more powerful... Check back latter.

Compatibility

The plug-in has been tested in the Servoy version 4.1.x under the java version 1.5+ & 1.6+ in windows, Macintosh & Ubuntu Linux 8.10 platform. The plug-in also working for Rich/Smart & Web Client.

Installation of the Plug-in & Sample Servoy Solution

Installation of mfServerUtils Plug-in

Installation of mfServerUtils is very easy. Please, follow the below steps to install the Server Utility Plug-in for Servoy, "mfServerUtils"

Steps:

- Download the Plug-in from [here](#) and unzip the file.
- Put all files(mfServerUtils.jar & mfServerUtils.jar.jnlp) in the <Servoy_Root_Directory> \application_server\plugins folder & Restart Servoy.
- Go to the Servoy Admin Page by navigating to http://server_ip:port/servoy-admin. Move on to the Server Plug-ins page, by clicking on the link "Server Plug-ins" at left menu.
- Now, you can able to see the "com.mindfire.plugins.mfserverutils.accesscode" property listed under the Header, "Mindfire Solutions: Server Utility Plug-in".
- Set a value for the access-code for the Plug-in. FOR SECURITY REASON, ACCESS-CODE MUST BE SET UP FROM THE SERVOY ADMIN PAGE & CALLED PRIOR TO EVERY CALL TO THE PLUG-IN. Restart Servoy Server.
- Now, you are ready with the Plug-in and its powerful features. For Installation of Sample Servoy Solution, check below.

Installation of Sample Servoy Solution

Please, follow the below steps to install the sample Servoy Solution. We recommend the mfServerUtils plug-in must be installed prior to the installation of Sample Solution.

Steps:

- Install mfServerUtils Plug-in. Ignore if already done.
- Download the Sample Servoy Solution from [here](#).
- Import the sample Servoy solution.
- Checkout the solution to your local drive.

- Navigate to the List of Global Variables of the solution, Change the value of “gAccessCode” global variable to your access-code setup from your Servoy Admin Page.
- Now, you are ready with the Sample Solution.

Run the Solution in Smart Client.

How to Use

You can get sample code for each method by doing a “move sample” for methods of the Plug-in.

Accessing Application Server Instance

You can get Instance of the Application Server by calling to the `getApplicationServer()` method of the Plug-in. Any calls to the plug-in must followed by checking the access-code. This will return you an object of class `JSApplicationServer`. Now, you can call any of the methods under the `JSApplicationServer` Node.

```
//Check for access-code
var appServer = null;
var access = plugins.mfServerUtils.checkAccessCode('<to_be_filled>');
if(access) {
    //Get Application Server Instance
    appServer = plugins.mfServerUtils.getApplicationServer();
}else {
    application.output('Oops!! access-code does not match.')}
}
```

You can get an instance of the Application Server running remotely by passing server address & the port number as arguments to the `getApplicationServer ()` method.

```
//Get Instance of an Application Server running remotely.
appServer = plugins.mfServerUtils.getApplicationServer("remote_ip","remote_port_number");
```

With the instance of the Application Server, you can shutdown the Application Server, alert all clients connected to the Application Server or restart Server.

```
//Alert all clients connected to the Application Server
appServer.alertAllClients('Hey, this message is from the server.');
```

```
//Shutting down the Application Server
appServer.shutdownServer();
```

```
//Restarting the Application Server
appServer.restartServer();
```

Accessing Client Instance

You can get the instance of a specific Client or instances of list of all connected Clients by calling to the `getClient()` or `getClients()` method respectively. Any calls to the plug-in must followed by checking the access-code. This will return you an object of class `JSClient`. Now, you can call any of the methods under the `JSClient` Node.

```

//Check for access-code
var access = plugins.mfServerUtils.checkAccessCode('<to_be_filled>');
if(access){

    //Getting a specific Client Instance
    var client = plugins.mfServerUtils.getClient(security.getClientID().toString());

    //Getting instance of All Clients connected to the Application Server.
    var clientList = plugins.mfServerUtils.getAllClients();

    for(var count=0; count<clientList.length; count++){
        var curClient = clientList[count];
        //Now, the curClient is single Client instance. Call any Client specific methods.
    }
}
else {
    application.output('Oops!! access-code doesn't match.')}
}

```

You can get an instance of the Clients running in the remote Application Server by passing server address & the port number as arguments to the getClient() or getClients() method.

```

//Getting a specific Client Instance of the Application Server running remotely
var client = plugins.mfServerUtils.getClient(security.getClientID().toString(),
                                             "remote_ip","remote_port_number");

//Getting instance of All Clients connected to the Application Server running remotely
var clientList = plugins.mfServerUtils.getAllClients("remote_ip","remote_port_number");

```

With the instance of Client, you can alert client, check the Client is alive or not or can shutdown clients.

```

//Alerting the Client
client.alertClient('Hey, you are alerted!!');

//Checking the Client is alive or not
if(client.isClientAlive()){
    application.output('Client is Alive');
}else{
    application.output('Client is not Alive');
}

//Shutting down Client
client.shutdownClient();

```

For a complete list of methods of the plug-in, have a look at the below section.

List of Properties & Methods of the Plug-in

The plug-in delivers the following properties & methods.

Plug-in Node

boolean checkAccessCode(String accessCode) :

Description: Used to check the access code being set previously by the application server (for Security Reason).

JSApplicationServer getApplicationServer() :

Description: Return you an instance of the Application Server.

JSClient getClient(String clientID) :

Description: Return you an instance of the Client of Client Id "clientID" running in the Application Server.

String getVersion() :

Description: Returns the current Plug-in version.

JSApplicationServer Node

void alertAllClients(String alertMsg) :

Description: Alert all the clients with the message "alertMsg" of the application server for which this object is an instance.

void restartServer() :

Description: Restart the Application Server for which this object is an instance.

void shutdownServer() :

Description: Shutdown the Application Server for which this is an instance.

JSClient Node

void alertClient(String alertMsg) :

Description: Alert the client with the message, alertMsg for which this object is an instance.

String getClientId() :

Description: Return the Client Id of the client for which this object is an instance.

boolean isClientAlive() :

Description: Returns a boolean value whether the client is alive for which this object is an instance.

void shutdownClient() :

Description: Shutdown the client session for which this object is an instance.