

**Oracle® Enterprise Single Sign-on
Provisioning Gateway**

.NET CLI SDK Guide

Release 11.1.1.1.0

E15695-01

March 2010

Copyright ©2005–2010 Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Table of Contents

Abbreviations and Terminology	4
About the ESSO-PG .NET CLI SDK	5
Installing .NET CLI	6
Using the .NET CLI as an SDK	7
Add a reference to the Passlogix.Provisioning.dll	7
Create an Instance of the IProvisioning Interface	7
Available Methods in IProvisioning Interface	9
Retrieving Results Using the IProvisioningResult Interface	16
ExtSearch Results	19
Sample Code (AddCredential)	24

Abbreviations and Terminology

Following is a list of commonly-used abbreviations and terminology.

Abbreviation or Terminology	Full Name
Administrative Console	ESSO-LM Administrative Console
Agent	ESSO-LM Agent
FTU	First Time Use Wizard
ESSO-AM	Oracle Enterprise Single Sign-on Authentication Manager
ESSO-Anywhere	Oracle Enterprise Single Sign-on Anywhere
ESSO-PG	Oracle Enterprise Single Sign-on Provisioning Gateway
ESSO-KM	Oracle Enterprise Single Sign-on Kiosk Manager
ESSO-LM	Oracle Enterprise Single Sign-on Logon Manager
ESSO-PR	Oracle Enterprise Single Sign-on Password Reset

About the ESSO-PG .NET CLI SDK

The .NET CLI SDK is provided with Oracle Enterprise Single Sign-on Provisioning Gateway (ESSO-PG). The SDK provides an interface for communicating with the ESSO-PG Web Service. These programming APIs live inside the assembly `Passlogix.Provisioning.dll`. This assembly leverages the main .NET CLI executable as an SDK library.

This guide is intended for experienced .NET application programmers responsible for the development of an organization's provisioning solutions.

Installing .NET CLI

The ESSO-PG .NET CLI must be installed prior to performing the steps in the document. Refer to the *ESSO-PG Installation and Setup Guide* for information on installing the ESSO-PG .NET CLI.

The .NET CLI is located under <Passlogix home>\v-GO PM\Client\DotNet.

Using the .NET CLI as an SDK

To use the .NET CLI as an SDK, complete the following steps:

1. In your .NET project, add a reference to the `Passlogix.Provisioning.dll`.
2. Create an instance of the `IProvisioning` interface.
3. Call the available methods on this interface (such as `AddCredential`, etc).
4. Use the returned `IProvisioningResult` interface to determine success and retrieve results.

Add a reference to the `Passlogix.Provisioning.dll`

Add a reference to `Passlogix.Provisioning.dll` in your .NET project:

1. From Visual Studio, load your solution and launch the **Solution Explorer**.
2. Select the applicable .NET project and expand it.
3. Right click on the **References** node and select **Add Reference**.
4. From the dialog, select **Browse** and find `Passlogix.Provisioning.dll` (can be found under `<Passlogix home>\v-GO PM\Client\dotnet`).
5. Click **Open**. A new reference to the assembly will be created.
6. Open the source file (with `.cs` extension) where the APIs will be called, and add the following lines to the top of the file:

```
using Passlogix.Provisioning;
using Passlogix.Provisioning.Exceptions;
```

Create an Instance of the `IProvisioning` Interface

In the same file, create a method to initialize an instance of the `IProvisioning` interface and add one of the following lines to that method:

```
// Method 1: If you know the full path

IProvisioning improv =

ProvisioningFactory.CreateFrom(@"<Path to .NET CLI>");

// Method 2: Load from same directory as provisioning assembly

IProvisioning improv = ProvisioningFactory.CreateFromPrivate();

// Method 3: To load file from the path (specified by %PATH%)

IProvisioning improv = ProvisioningFactory.CreateFromPath();
```

After you have selected a method for loading, check for errors and then set the credentials for connection to the ESSO-PG service:

```
// Code to use after method of loading assembly has been selected

if (improv != null)
{
```

```

try
{
    // You'll first need to establish a connection
    // or else all resulting calls to the methods will
    // fail. This method sets credentials for connecting
    // to PM service. It does not actually connect to
    // the service until a provisioning request is made.
    // You can connect in three ways:
    improv.Connect("Administrator", "password");
    // Assumes http://localhost/v-go pm service/up.asmx
    // and %COMPUTERNAME% is the Agent name.

    // Method 2 allows you to specify URL and Agent name
    improv.Connect(
        "http://<server>/v-go pm service/up.asmx",
        "My Agent",
        "Administrator", "password");
    // Method 3 allows you to specify URL.
    // This method is preferred since the web service
    // is not local but the user does not necessarily
    // want to specify the agent name (defaults to
    // %COMPUTERNAME%).
    improv.Connect(
        "http://<server>/v-go pm service/up.asmx",
        "Administrator", "password");
    // Make provisioning requests via the improv interface
    // Examples of this are given later in this document
    . . .
}
catch (ProvisioningException ex)
{

```



```
// Handle exception
}

}
```

After the connection has executed successfully, requests can be sent to the ESSO-PG Web service through the methods of the `ipro` variable. Each method returns its results in an `IProvisioningnResult` interface. Oracle recommends these methods be called within a `try...catch` block for error handling. Catching the `ProvisioningException` class is sufficient for any exceptions thrown by the CLI. Other exceptions can be handled by adding a catch (`Exception`) block.

Available Methods in IProvisioning Interface

This section lists all the available methods and their parameters for each provisioning operation. The following information is provided for each available method:

- Method name and description
- Method Overload List
- A description of the method's parameters (if applicable)

One of these parameters requires a special explanation. The `options` parameter is a dictionary of key-value pairs. The key is the name of the argument used by the CLI on the command line. The value is its value. The developer can set a key-value pair in the dictionary using either the literal name of the key (passed on the command line) or the key constants defined in the `OperationKeys` class.

- Command-line syntax used by the CLI (`CLI_Syntax`) (if applicable)

The command-line arguments map directly to the valid keys that can be used to fill the `options` parameter of a method. The `OperationKeys` class has been provided for convenience with constants mapping to the literal value of each key. This can be used to fill or index the `options` array. For brevity, the CLI Syntax does not show the full syntax. Refer to the *ESSO-PG CLI Guide* for full syntax information. The operation name is capitalized. Arguments specified in brackets are optional.

Method	Description
<code>Connect</code>	Establishes connection to Web service. This method does not actually attempt the connection but stores the credentials used to connect for use by other methods.

Overload List

```
void Connect(string strUsername, string strPassword);

void Connect(string strURL, string strUsername, string
strPassword);

void Connect(
    string strURL,
    string strAgent,
    string strUsername,
    string strPassword);
```

Parameter	Description
<code>strURL</code>	Web Service URL. Default is http://localhost/v-GO%20PM%20Service/up.aspx
<code>strAgent</code>	Identifier for this agent. Default is %COMPUTERNAME%.
<code>strUsername</code>	Username used to authenticate against the Web service.
<code>strPassword</code>	Password used to authenticate against the Web service.

Method	Description
<code>SetExecTime</code>	Sets the execution time of the provisioning instruction. This can be used to tell the instruction to execute in the agent at a future date or time after it has been created. If this is not set, it defaults to "Now."

Overload List

```
void SetExecTime(DateTime dtExec);
```

Method	Description
<code>AddCredential</code>	Provision the user with a new credential.

Overload List

```
IProvisioningResult AddCredential(
    string strUserId,
    string strApplication,
    string strDescription,
    string strAppUserId,
    string strPassword);
IProvisioningResult AddCredential(
    string strUserId,
    string strApplication,
    StringDictionary options);
```

Parameter	Description
<code>strUserId</code>	User ID of user to be provisioned.
<code>strApplication</code>	Name of the application to provision.
<code>strDescription</code>	Description of the provisioning instruction.
<code>strAppUserId</code>	Application user ID of the credential.

<code>strPassword</code>	Password of the credential.
<code>options</code>	Hashtable of options (keys specified by <code>OperationKeys</code>).

CLI Syntax

```
ADD_CREDENTIAL sso_userid sso_application [sso_app_userid]
sso_password] [sso_description] [sso_other1] [sso_other2]
```

Method	Description
<code>CancelRequest</code>	Cancel the provisioning request (before the agent runs).

Overload List

```
IProvisioningResult CancelRequest(string strUserId, string
strGuid);
```

Parameter	Description
<code>strUserId</code>	User ID of user to be provisioned.
<code>strGuid</code>	ID of provisioning instruction to cancel (returned by several methods) that can be canceled.

CLI Syntax

```
CANCEL sso_userid=<username> command_id=<guid>
```

Method	Description
<code>DeleteCredential</code>	Delete a provisioned credential.

Overload List

```
IProvisioningResult DeleteCredential(string strUserId,
string strApplication, string strAppUserId, string strOther1,
string strOther2);
```

```
IProvisioningResult DeleteCredential(string strUserId,
string strApplication, StringDictionary options);
```

Parameter	Description
<code>strUserId</code>	User ID of user to be provisioned.
<code>strApplication</code>	Name of the application to provision.
<code>strAppUserId</code>	Application User ID of the credential.

strOther1	Other field value (1).
strOther2	Other field value (2).
options	Hashtable of options (keys specified by OperationKeys).

CLI Syntax

```
DELETE_CREDENTIAL sso_userid sso_application [sso_app_userid]
[sso_password] [sso_other1] [sso_other2]
```

Method	Description
ModifyCredential	Modify a provisioned credential.

Overload List

```
IProvisioningResult ModifyCredential(string strUserId,
    string strApplication, string strAppUserId,
    string strDescription, string strPassword, string strOther1,
    string strOther2);
IProvisioningResult ModifyCredential(string strUserId,
    string strApplication, string strAppUserId,
    StringDictionary options);
```

Parameter	Description
strUserId	User ID of user to modify.
strApplication	Name of the application of credential to modify.
strAppUserId	Application User ID of the credential to modify.
strAppUserId	Password of the credential to modify.
strDescription	Description of the provisioning instruction.
strOther1	Other field value (1).
strOther2	Other field value (2).
options	Hashtable of options (keys specified by OperationKeys).

CLI Syntax

```
MODIFY_CREDENTIAL sso_userid sso_application sso_app_userid
```

```
[sso_description] [sso_password] [sso_other1] [sso_other2]
[sso_password] [sso_other1] [sso_other2]
```

Method	Description
DeleteUser	Delete the user container (similar to deleting all credentials for a particular user).

Overload List

```
IProvisioningResult DeleteUser(string strUserId);
```

Parameter	Parameter
strUserId	User ID of container to delete.

CLI Syntax

```
DELETE_USER sso_userid=<username>
```

Method	Description
GetStatus	Ping the server. If it returns successfully without error, the server is functioning.

Overload List

```
IProvisioningResult StatusRequest(string strUserId, string strGuid);
```

CLI Syntax

```
STATUS sso_userid=<username> command_id=<guid>
```

Method	Description
StatusRequest	Request the status of a pending provisioning instruction.

Overload List

```
IProvisioningResult StatusRequest(string strUserId, string strGuid);
```

Parameter	Parameter
strUserId	User ID to query.
strGuid	ID of provisioning instruction (returned by several methods)

CLI Syntax

```
STATUS sso_userid=<username> command_id=<guid>
```

Method

GetSettings

Description

Return the directory settings of the PM Web service.

Overload

List<https://passportal.passlogix.com/Passlogix%20Documentation/Forms/AllItems.aspx>

```
IProvisioningResult GetSettings();
```

CLI Syntax

```
GET_SETTINGS
```

Method

Description

GetSchema

Get the schema (or list of available options for SetSettings).

Overload List

```
IProvisioningResult GetSchema();
```

CLI Syntax

```
CLI Syntax: GET_SCHEMA
```

Method

Description

SetSettings

Change the settings used by the Web service.

Overload List

```
IProvisioningResult SetSettings(IDictionary map).
```

Parameter

Description

Map

Key-value pair for each setting.

CLI Syntax

```
SET_SETTINGS name="key1, key2, ..." value="value1, value2, ..."
```

Method

Description

ExtSearch

Search the directory service and return information on users, applications, logs. This returns a list of applications that can be provisioned for a particular user or all users.

Overload List for Applications

```
IProvisioningResult ExtSearchApplications();
```

```
IProvisioningResult ExtSearchApplications(string strUserId);
```

Parameter	Description
<code>strUserId</code>	Name of user whose application list should be returned.

Overload List for Users

```
IProvisioningResult ExtSearchUsers(); IProvisioningResult
ExtSearchUsers(string strUserId,
    StringCollection logons, bool fRetLogons, bool fRetInsts,
    bool fMatchExact);
IProvisioningResult ExtSearchUsers(StringDictionary options);
```

Parameter	Description
<code>strUserId</code>	User to return information on.
<code>logons</code>	Return only these logons (csv format).
<code>fRetLogons</code>	Return logon information.
<code>fRetInsts</code>	Return pending provisioning instructions.
<code>fMatchExact</code>	Use exact match on <code>strUserId</code> .
<code>options</code>	Hashtable of options (specified by <code>ExtSearchKeys</code>).

Overload List for Logging

```
IProvisioningResult ExtSearchLog();
IProvisioningResult ExtSearchLog(EventType evt);
IProvisioningResult ExtSearchLog(DateTime dtStart, DateTime
dtEnd,
    EventType evt);
```

Parameter	Description
<code>evt</code>	EventType to return.
<code>dtStart</code>	Start date of range to return.
<code>dtEnd</code>	End date of range to return.

CLI Syntax

```
EXT_SEARCH CATALOG=Applications [userId=<username>]
EXT_SEARCH CATALOG=Users [userId=<username>]
[logon="logon1,logon2,..."] [returnLogons=true|false]
[returnInstructions=true|false] [uidMatch=substring|equal]
```

```
EXT_SEARCH CATALOG=EventLog [startDate=mm/dd/yyyy]
[endDate=mm/dd/yyyy]
[eventType=amducs]
```

Retrieving Results Using the IProvisioningResult Interface

After a provisioning request to the ESSO-PG Web Service has completed, an `IProvisioningResult` interface is returned by the called method. Your application can use this interface to determine whether if the request has completed successfully and retrieve any relevant results. This section shows the available properties on the `IProvisioningResult` interface and how to interpret their values for the methods called from `IProvisioning`.

Interface Definition

```
public interface IProvisioningResult
{
    string Response
    {
        get;
    }

    bool Success
    {
        get;
    }

    string CommandID
    {
        get;
    }

    string ErrorMessage
    {
        get;
    }
}
```



```
IDictionary AttributesCollection
{
    get;
}
}
```

Property	Description
Success	True if the command completed successfully.
ErrorMessage	The error string if Success is False. May not always be set.
CommandID	The unique ID associated with the completed command (a 32-digit GUID)). All methods except <code>ExtSearch</code> return a GUID. However, only the following methods provide a GUID that can be used by the <code>CancelRequest</code> and <code>StatusRequest</code> operation: <ul style="list-style-type: none"> • <code>AddCredential</code> • <code>ModifyCredential</code> • <code>DeleteCredential</code>
Response	The raw XML response returned by Web service. This is useful if the results need to be re-parsed.
AttributesCollection	Detailed results returned by Web service on Success. The format is a Dictionary of key-value pairs. The methods that fill this property are: <ul style="list-style-type: none"> • <code>GetSettings</code> • <code>GetSchema</code> • <code>StatusRequest</code> • <code>ExtSearch</code>

AttributesCollection

This is a dictionary collection of attributes returned by `GetSettings`, `GetSchema`, `ExtSearch`, and `StatusRequest`. The keys are strings that represent the attribute name. The values can either refer to another `IDictionary`, an `ICollection`, or a string. However, types are not mixed within the same collection. After the type has been established, the same type is referenced by all keys.

The following table describes the meaning of the keys and values returned by the provisioning operations listed:

Methods	Description
<code>GetSettings</code>	Returns a collection of string key-value pairs. The key is the name of the setting. The value is its value. These are the storage values set in

the registry by the ESSO-PG Web Service.

`StatusRequest` Returns a collection of string key-value pairs. The *key* is the name of a status property. The *value* is its value. The following status keys are supported:

Status Key	Value
InstructionState	PENDING, PROCESSED
Result	SUCCESS, FAILED
Description	SUCCESS, <Reason for failure>
Modified	<Date modified>

`GetSchema` The *key* is a string that represents the name of a group of storage settings. The value is an `IList`. Each `IList` entry describes one setting under this group. The entry is an `IDictionary` of string key-value pairs. The key can be one of the following followed by one of the possible values:

Key	Value
DataType	Can be string or bool
DisplayDesc	A description of this setting. Can be empty.
DisplayName	The friendly name of this setting to display.
Flags	An internal value used to describe if the settings is non-persistent, must exist
RegDefault	The default value for this setting. Can be empty.

	RegName	The name of the registry key.
	RegPath	The relative registry path to this setting.
	RegType	The registry type (DWORD or string).
	[note] The setting described by this entry becomes a value that can be retrieved or set by <code>GetSettings</code> and <code>SetSettings</code> .	
ExtSearch	Collection of hashtables. (See next section for more information). The key is a string but the type of the returned value depends on the <code>ExtSearchXXX</code> called.	



The structure and format of the returned key-value pairs from the `AttributesCollection` property are designed to closely mirror the console output from the actual CLI. Simply using the CLI will help in understanding the format and structure of the collection returned by these methods.

ExtSearch Results

This section describes the format of the `AttributesCollection` map returned by `ExtSearch`.

ExtSearchApplications

Returns:

.NET: `HashTable of HashTables`

Java: `HashMap of HashMaps`

Key	Value						
Application Name	HashTable (string key/value pairs)						
	<table><tr><th>Key</th><th>Value</th></tr><tr><td>HasFourthField</td><td>True False</td></tr><tr><td>HasPassword</td><td>True False</td></tr></table>	Key	Value	HasFourthField	True False	HasPassword	True False
Key	Value						
HasFourthField	True False						
HasPassword	True False						

HasThirdField	True False
HasUserId	True False
IsSecurId	True False

If IsSecurId is true, then the first four fields are renamed:

- SecurID-UserId
- SecurID-Other[4th]
- HasPassword
- PassKeyType

Adobe Acrobat Reader

```
HasFourthField: False
HasPassword: True
HasThirdField: False
IsSecurID: False
HasUserId: False
```

MSN Messenger

```
HasFourthField: False
HasPassword: True
HasThirdField: False
IsSecurID: False
HasUserId: True
```

Visual SourceSafe

```
HasFourthField: False
HasPassword: True
HasThirdField: True
IsSecurID: False
HasUserId: True Users
```

Returns:

.NET: HashTable of Lists of HashTables

Java: HashMap of Lisis of HashMaps

Key	Value
User's Name	
Logon Entry	
Key	Value
name	Application name
modifiedDate	Date last modified
lastUsedData	Date last used by SSO
Id	GUID identifier
Pending Entry	
applicationName	Application
createDate	Date created
executeDate	Date this will execute
id	GUID identifier
GUID identifier	ADD MODIFY DELETE
provisioningAgent	Agent name
status	SUCCESS Pending

CLI Output:

```
ext_search catalog=users returnLogons=true
```

This returns a list of logons for all users.

johnd

```
modifiedDate: 2005-08-24 16:43:41Z
lastUsedDate: 2005-08-24 16:43:41Z
name: Adobe Acrobat Reader
id: a75f58c8-a3bd-4d00-bc27-99a587dd98f8
```

```
modifiedDate: 2005-08-24 16:43:41Z
lastUsedDate: 2005-08-24 16:43:41Z
name: Adobe Acrobat Reader
id: d6bc375d-3f90-400b-a012-6b80aff4ef49
```

```
modifiedDate: 2005-09-09 16:28:15Z
lastUsedDate: 2005-09-09 16:28:15Z
name: Visual SourceSafe
id: 80cdc929-61a6-4b86-8763-d5f02b0dbb8b
```

```
modifiedDate: 2005-09-01 17:30:26Z
lastUsedDate: 2005-09-01 17:30:26Z
name: Visual SourceSafe
id: 065f5cff-b651-4a3a-a99c-c606059cbad7
```

```
modifiedDate: 2005-09-09 16:41:33Z
lastUsedDate: 2005-09-09 16:41:33Z
name: Visual SourceSafe
id: 0a0686b5-3e38-4830-8e02-79b8177de0b4
```

ExtSearchLog

Returns:

.NET: HashTable of HashTables

Java: HashMap of HashMaps

Key

Value

Entry Number

HashTable (string key/value pairs)

Key

Value

applicationName

Application name

eventType

Type of event (DWORD flag)

executeDate

Date executed

id

GUID identifier

provisionedUser

User provisioned

provisioningAgent

Agent name

timeStamp

Time stamp

CLI Output:

```
ext_search catalog=eventLog
```

This returns a list of logons for all users.

Entry 1

```
applicationName:
eventType: 64
executeDate: 0001-01-01 00:00:00.000Z
id: a09b9de7-4b65-464c-8dc8-90219e222991
provisionedUser:
provisioningAgent: SSO PM Console
timestamp: 2005-11-17 18:33:37.290Z
```

Entry 2

```
applicationName:
eventType: 64
executeDate: 0001-01-01 00:00:00.000Z
id: bd444f6c-e3cf-4efc-bbd8-c5e82d55ed96
provisionedUser:
provisioningAgent: SSO PM Console
timestamp: 2005-11-17 18:33:37.370Z
```

Entry 3

```
applicationName:
eventType: 64
executeDate: 0001-01-01 00:00:00.000Z
id: 6eebd1dd-a904-43db-8c22-38ef941e83b3
provisionedUser:
provisioningAgent: SSO PM Console
timestamp: 2005-11-17 18:33:38.960Z
```

Entry 4

```
applicationName: Visual SourceSafe
eventType: 4
executeDate: 2005-11-17 19:28:51.427Z
id: 2c45f078-c9c7-4268-9abd-4e50111ba644
```

```
provisionedUser: davidh
provisioningAgent: SSO PM Console
timestamp: 2005-11-17 19:28:51.427Z
```

Sample Code (AddCredential)

The following code demonstrates how to call the `AddCredential` method from the `IProvisioning` interface. This example demonstrates adding a credential for the ESSO-LM user "johndoe". The application being added is Yahoo and the credentials for this application are "jdoe" and "password." The description of this credential is "Test App."

```
try
{
    IProvisioningResult ipr = improv.AddCredential(
        "johndoe",
        "Yahoo",
        "Test App",
        "jdoe",
        "password");
    // Process results in ipr
    if (!ipr.Success)
    {
        Console.WriteLine(ipr.ErrorMessage);
        return;
    }
    // Display GUID
    Console.WriteLine("SUCCESS" + ipr.CommandID);
}
catch (ProvisioningException ex)
{
    // Handle Exception...
}
```

Credentials can also be added using an options argument, which is a more flexible method of passing. This method allows the use of additional parameters (some applications require an OTHER1 and OTHER2 field) and their combinations:

```
StringDictionary options = new StringDictionary();
options.Add(OperationKeys.DESRIPTION, "Test App");
options.Add(OperationKeys.APP_USERID, "jdoe");
```



```
options.Add(OperationKeys.PASSWORD, "password");  
  
options.Add(OperationKeys.OTHER1, "VGO");  
  
IProvisioningResult ipr = improv.AddCredential("johndoe",  
"Visual SourceSafe", options);
```

This example demonstrates how to add a credential for the “Visual SourceSafe” application for the SSO user “johndoe”. Since this application requires an OTHER1 field, this method is the only way to add the credential.