MICRO FOCUS

SecureLogin 8.8

Java API Guide

December 2019

Legal Notice

For information about legal notices, trademarks, disclaimers, warranties, export and other use restrictions, U.S. Government rights, patent policy, and FIPS compliance, see https://www.microfocus.com/about/legal/.

Copyright © 2019 Micro Focus. All rights reserved.

Contents

About This Guide 5

1 API Overview 7

2 Configuring the User Data 9

3 Configuring the User Directory 11

4 Handling the Exceptions 13

5 Using the Command Line to Configure the Java APIs 15

Provisioning the User Credentials 16
Linking the User Credentials with the Application 17
Provisioning Passphrase for the User 17
Disabling the Passphrase Prompt 18

About This Guide

This guide describes the Java APIs supported by SecureLogin. It includes step-by-step instructions for using these APIs.

Intended Audience

This guide is intended for SecureLogin administrators, developers and consultants who are creating custom provisioning of applications with SecureLogin. It is assumed that you have the basic knowledge of Java.

Additional Documentation

For the most recent version of this guide and other SecureLogin documentation resources, see the SecureLogin Documentation (https://www.netiq.com/documentation/securelogin) and keep up to date on patches and versions of both SecureLogin and the host operating system.

Contact Information

We want to hear your comments and suggestions about this book and the other documentation included with this product. You can use the **comment on this topic** link at the bottom of each page of the online documentation, or send an email to Documentation-Feedback@microfocus.com.

For specific product issues, contact Micro Focus Customer Care at https://www.microfocus.com/support-and-services/.

1 API Overview

The following scenarios explains the use cases where you can make use of this API in your deployment scenario:

Scenario 1: Provisioning the application credentials of a new user in the SecureLogin data to allow access to all the required applications.

Scenario 2: Provisioning the legacy application credentials in the SecureLogin data to enable identity governance.

You can configure the scenarios mentioned above using the command line interface. For more information on how to use command line to configure API, see Chapter 5, “Using the Command Line to Configure the Java APIs,” on page 15.

The SecureLogin Java single sign-on API includes the following components:

- **UserAPI:** It allows provisioning and de-provisioning of user key store master secrets and the application credentials. See Chapter 2, “Configuring the User Data,” on page 9.
- **SchlapiConfig:** It allows you to customize the your configuration. Use `SchlapiConfig` to obtain the directory configurations or to store certificates. See Chapter 3, “Configuring the User Directory,” on page 11.
- **SchlapiException:** `UserAPI` uses `SchlapiException` as a general purpose exception class to report errors. See Chapter 3, “Configuring the User Directory,” on page 11.

2 Configuring the User Data

An administrator can use UserAPI to allow Java applications to integrate with the SecureLogin single sign-on data. The UserAPI class can perform the following actions.

- Validating the passphrase answer against the passphrase question
- Provisioning the user credentials

A UserAPI instance is created using the SchlAPI interface. The SchlAPI interface defines how the UserAPI and supporting classes connect to the directory and SchlAPI license key. Once the UserAPI instance is created, attach it to a user object by directory DN before you invoke other methods. For example:

```
UserAPI api = new UserAPI(new MyConfig());

api.attach("cn=myname,cn=users,dc=protocom,dc=com");

String question = api.getQuestion();
```

Constructor Details

The following table includes the constructor details to create an UserAPI instance.

Constructor	Description
`UserAPI(SchlapiConfig config)`	Creates an UserAPI instance.

The UserAPI instances are not locked to an object after attach in invoked. To access another LDAP object, invoke attach with that object's DN.

Methods Details

The following table explains the methods and modifiers to use with UserAPI.

Modifier and Type	Method and Description
boolean	`attach(String object)` Attaches this instance to an LDAP object.
void	`close()` This is not part of attach but must be invoked when you no longer need the UserAPI instance so that LDAP connections can be closed.
void	`deprovisionAccount(String account)` Removes an account from the directory.

Modifier and Type	Method and Description
`string`	`getQuestion()` Displays the SecureLogin single sign-on passphrase question.
`void`	`provisionAccount(String account, Map credentials, String description)` Creates an account and provisions credentials.
`void`	`provisionLinks(String platform, String credId, String isSetPlat)` Links credentials to a specific application.
`void`	`provisionPassphraseQA(String password, String question, String answer)` Provisions an account with a new passphrase question and answer.
`void`	`removeNonRepudiation(String password, String answer)` Re-encrypts the user's entries using the new Windows password.
`boolean`	`verifyAnswer(String answer)` Indicates if the answer to the SecureLogin single sign-on question is correct.

3 Configuring the User Directory

The `SchlapiConfig` class allows the Java applications to integrate with the directory that stores the SecureLogin single sign-on data. A user with necessary right can access the directory using the `UserAPI` with `SchlAPIConfig`. The following is an example of credentials provisioning.

```
Map creds = new HashMap();
  creds.put("username", "Joe User");
  creds.put("password", "my secret");
  SchlapiConfig config = new SchlapiConfig() { ... }; // API user's will
need to provide details of the directory configuration
  UserAPI api = new UserAPI(config);
  api.attach("cn=juser,ou=engineering,o=corporation");
  api.provisionAccount("some account", creds, null);
  api.close();
```

Methods Details

The following table explains the methods and modifiers to use with `SchlAPIConfig`.

Modifier and Type	Method and Description
`string`	`getCertificateFile()` If you are using SSL you have two options for providing the server certificate. You can either import it into Java's keystore as a trusted certificate or you can specify the certificate filename. It requires JRE v1.4 or above. Specify null if you do not wish to provide a certificate.
`string []`	`getContexts()` Contexts to search for users if the full LDAP DN is not provided.
`string`	`getLicense()` This is the license key provided by Protocom.
`string`	`getPassword()` Password for the user specified in `getUser()`, this password is used to connect to the directory.

Modifier and Type	Method and Description
`int`	`getPort()` Port number to connect to the directory. This is usually 389 (LDAP) or 636 (LDAP/SSL).
`string`	`getServer()` The IP address or DNS name of the directory server that includes single sign-on data.
`string`	`getUser()` Username to connect to the directory. `UserAPI` requires a user with required rights to the SecureLogin single sign-on attributes.
`boolean`	`useSSL()` Indicates to use SSL for connection.

4 Handling the Exceptions

`UserAPI` uses the `SchlapiException` class as a general purpose exception to report the error messages.

5 Using the Command Line to Configure the Java APIs

This section describes how to configure the API using the command line. The API configuration parameters are provided directly from the command line. The following table lists the Windows Installer command line options used to perform various configurations required to login to an application using SecureLogin.

IMPORTANT: You must specify the `admin.json` file while using the following arguments except `-h` and `--help`.

Table 5-1 *Java API Command Line Options*

Arguments	Usage
`-a`	Displays administration details.
`-p, --provision`	Provisions credential to a user, group or container.
`-l, --link`	Links credential to an application for a user, group or container.
`-pp, --pprovision`	Provisions passphrase for a user.
`-r`	Re-encrypts the SecureLogin data with the new password.
`-h, --help`	Displays the API help.

The above mentioned command line arguments are used with the `admin.json` file and the `user.json` file. The `admin.json` file contains the administrator details and the `user.json` file contains the user credentials for the application and the passphrase. The following are the examples of these files.

Admin.json

```
"user":"CN=Administrator,CN=Users,DC=domain,DC=com",
  "password":"samplepassword",
  "server":"WIN-O8QA8KVQ9OH.domain.com",
  "SecurePort":"636",
  "CertPath":"C:\\Users\\username\\Documents\\NSL Provisioning SDK-8.8.0-
0\\dp3.cer",

"licence":"3266238b8bcbd0b5e8ad8e552ef97fc44ca64dc5b20c588c9c1c2e9d0026863
31841d66a09ce66b1"
```

User.json

```
{
  "users":[
    {
      "dn":"cn=user,ou=users,o=data",
      "password":"samplepassword",
      "credentials":[
          {
            "credential_name":"samplecred",
            "details": {
              "Username":"sampleuser",
              "password":"samplepass"
            }
          }
      ],
      "link":[
          {
            "credential_name":"samplecred",
            "application_name":"sampleapp.exe"
          }
        ],
      "passphrase":{
        "question":"What is the name of your organisation",
        "answer":"Micro Focus"
      }
    }
  ]
}
```

You can create your own `.json` files or edit the files provided. Provisioning an application includes the following configurations. You must download the Java Single Sign-on package and extract the `Java SSO SDK-8.8.0-0.zip` file before you start performing the following configurations.

- “Provisioning the User Credentials” on page 16
- “Linking the User Credentials with the Application” on page 17
- “Provisioning Passphrase for the User” on page 17
- “Disabling the Passphrase Prompt” on page 18

Provisioning the User Credentials

Perform the following steps to create credentials that allow user to login to an application using SecureLogin.

1. Navigate to the folder where the JSSO API files are saved.
2. Open the command prompt and run the following command:

   ```
   java -jar "jssoapi-cmd-8.8.0-0.jar" -a admin.json -p user.json
   ```

   NOTE: The `user.json` file must contain the following information.

```
"credentials":[
          {
            "credential_name":"samplecred",
            "details": {
              "Username":"sampleuser",
              "password":"samplepass"
            }
          }
      ]
```

3 Restart SecureLogin.

4 Verify that the user credentials created in Step 2 are listed in **SecureLogin > My Logins**.

Linking the User Credentials with the Application

Perform the following steps to link the user credentials with the application.

1 Navigate to the folder where the JSSO API files are saved.

2 Open the command prompt and run the following command:

```
java -jar "jssoapi-cmd-8.8.0-0.jar" -a admin.json -l user.json
```

NOTE: The `user.json` file must contain the following information.

```
"credentials":[
          {
            "credential_name":"samplecred",
            "details": {
              "Username":"sampleuser",
              "password":"samplepass"
            }
          }
      ],
"link":[
          {
            "credential_name":"samplecred",
            "application_name":"sampleapp.exe"
          }
        ]
```

3 Restart SecureLogin or perform a cache refresh to refresh the data store settings.

Provisioning Passphrase for the User

Perform the following steps to provision the passphrase question and answer for a user.

NOTE: If the passphrase is already provisioned, performing these steps will change the passphrase.

1 Navigate to the folder where the JSSO API files are stored.

2 Open the command prompt and run the following command:

```
java -jar "jssoapi-cmd-8.8.0-0.jar" -a admin.json -pp user.json
```

NOTE: The `user.json` file must contain the following information.

```
"passphrase":{
        "question":"What is the name of your organisation",
        "answer":"Micro Focus"
```

3 Restart SecureLogin.

4 Navigate to **SecureLogin > Advanced > Change Passphrase** and verify that the passphrase is provisioned.

Disabling the Passphrase Prompt

When password is changed, SecureLogin prompts the user to specify passphrase answer. You can configure SecureLogin to re-encrypt the users' data using the new password without prompting for the passphrase. Perform the following steps to disable the passphrase prompt:

1 Navigate to the folder where the JSSO API files are stored.

2 Open the command prompt and run the following command:

```
java -jar "jssoapi-cmd-8.8.0-0.jar" -a admin.json -r user.json
```

NOTE: The `user.json` file must contain the new password in the following format before you run this command.

```
"users":[
    {
      "dn":"cn=user101,ou=users,o=data",
      "password":"newpassword"
```

3 Restart SecureLogin.