**PUBLIC**
SAP Single Sign-On 3.0 SP02
Document Version: 1.2 – 2022-08-26

# One-Time Password Authentication Implementation Guide



THE BEST RUN SAP

# Content

# 1 One-Time Password Authentication Implementation Guide

The one-time password (OTP) solution, part of SAP Single Sign-On (SSO), is used to generate one-time passwords called passcodes.

The passcodes are time-based and valid for just one login attempt. They are used for strong authentication when logging on to corporate resources. To use one-time password authentication, it is best to have compatible mobile devices, which provide the passcode. To use the OTP authentication, you should then install an authenticator application on your mobile devices.

> **i** Note
>
> If you do not have compatible mobile devices, there are alternative solutions for passcode provisioning.

You can use the OTP authentication in the following cases:

- Log on to systems using Secure Login Client.
- Log on to systems using SAML and using SAP NetWeaver (NW) Application Server (AS) Java as an identity provider.
- Log on to web applications running on SAP NetWeaver AS for Java.

The OTP solution provides its own login module, `TOTPLoginModule`, which supports **two-factor authentication** or **single-factor authentication**. With two-factor authentication, you provide two means of identification (a passcode and a corporate password or an X.509 certificate for example), while with single-factor authentication you log on using only one factor (a passcode, a corporate password, an X.509 certificate or other).

## Related Information

One-Time Password Authentication Administration Guide [page 5]
One-Time Password Authentication User Guide [page 95]
One-Time Password Authentication Developer Guide [page 55]
One-Time Password Authentication Security Guide [page 103]
Troubleshooting [page 106]

## 1.1 One-Time Password Authentication Installation and Upgrade Guide

This section explains the requirements and the procedure for the installation or upgrade of SSO AUTHENTICATION LIBRARY 3.0.

### 1.1.1 System Requirements

You can install SSO AUTHENTICATION LIBRARY 3.0 on SAP Netweaver Application Server (AS) Java 7.30 or higher.

### 1.1.2 Installation Steps

To install SSO AUTHENTICATION LIBRARY 3.0, perform the following steps:

#### Procedure

1. Go to the SAP Software Download Center
2. Choose *Support Packages and Patches* .
3. In the A-Z alphabetical list, navigate to the S section.
4. Navigate to the following product: *SAP SINGLE SIGN ON* > *SAP SINGLE SIGN-ON 3.0* > *Comprised Software Component Versions* > *SSO AUTHENTICATION LIBRARY 3.0*
5. Download SSOAUTHLIB <release>.sca.
6. Deploy the SCA to the AS Java.

### 1.1.3 Upgrade Steps

If you have used the *One-Time Password Administration* UI of SAP Single Sign-On version 2.0 SP06 or lower, you will still be able to create policy scripts in the *One-Time Password Administration* UI until you migrate. To benefit from the centralized management and versioning of policy scripts, we recommend you to migrate your scripts to the new *Policy Script Administration Console*. In the *One-Time Password Administration* UI you will be prompted to migrate your policy scripts.

You have the following options:

- If you ignore the migration, you can continue to work with the policy scripts as in version SAP Single Sign-On 2.0. For more information, see Access Policies Implementation Guide.

- If you migrate, the existing policy scripts will be transferred to the *Policy Script Administration Console*, where you can create and manage new policy script versions. For more information, see Policy Scripts Implementation Guide.

## 1.2 One-Time Password Authentication Administration Guide

This section provides information about the installation and configuration of one-time password (OTP) authentication.

### Types of Passcodes

To use OTP authentication, you have to configure how the one-time password (passcode) is generated and provided to your users. You can configure authentication with the following types of passcodes:

- Passcode generated by an authenticator mobile application
  This configuration is possible if users can install an authenticator application on their mobile devices. For more information, see Configuring Authentication with a Passcode Generated by an Authenticator App [page 32]
- Random passcode sent by SMS, e-mail, or another channel
  This configuration is for users who cannot install an authenticator mobile application. The passcode can be sent to the users by SMS or e-mail. For more information about this configuration, see Configuring Authentication with a Random Passcode Sent by SMS, E-Mail, or Another Channel [page 32].
- External passcode generated by a third-party passcode provider
  This configuration is for passcodes from a third-party provider (for example, RSA SecureID passcodes) that need to be validated before the user is authenticated. For authentication with these passcodes, developers have to implement a policy script written in JavaScript. For more information about this configuration, see Configuring External Passcode Validation [page 35].

### Operations in SAP NetWeaver Administrator

You access SAP NetWeaver Administrator at `http://<host>:<port>/nwa` and can use it for the following operations:

| Operation | Reference |
| --- | --- |
| Setting OTP-related login modules | Configuring TOTPLoginModule and RBALoginModule [page 8] |
| Specifying OTP roles, groups, or users | Assigning Principals to UME Roles or Groups |

| Operation | Reference |
| --- | --- |
| Configuring passcode provisioning for users with unsupported mobile devices | Configuring Authentication with a Random Passcode Sent by SMS, E-Mail, or Another Channel [page 32] |
| Configuring the use of external passcodes | Configuring External Passcode Validation [page 35] |
| Setting an OTP-related logon application | Configuring an OTP-Related Logon Application [page 38] |
| Getting OTP logs and traces | Collecting Traces with the Security Troubleshooting Wizard [page 111] |
| Configuring logon to supported systems | Configuring Logon to Supported Systems [page 25] |

## Operations in One-Time Password Administration UI

You access the *One-Time Password Administration* UI at `http://<host>:<port>/ssoadmin/otp` and can use it for the following:

| Operation | Reference |
| --- | --- |
| Configuring specifics for single-factor authentication | Configuring Single-Factor Authentication [page 28] |
| Configuring specifics for two-factor authentication | Configuring Two-Factor Authentication [page 30] |
| Enabling users to do online account setup | Configuring the Online Account Setup [page 41] |
| Disabling user accounts | Managing User Accounts [page 43] |
| Setting a policy for locking of user accounts | Additional Settings [page 51] |
| Unlocking user accounts | Managing User Accounts [page 43] |
| Setting the validity of user accounts | Managing User Accounts [page 43] |
| Advanced user search | Managing User Accounts [page 43] |
| Setting passcode length | Additional Settings [page 51] |
| Setting passcode digest algorithm | Additional Settings [page 51] |
| Configuring passcode provisioning for users with unsupported mobile devices | Configuring Authentication with a Random Passcode Sent by SMS, E-Mail, or Another Channel [page 32] |
| Configuring the use of external passcodes | Configuring External Passcode Validation [page 35] |
| Customizing the *Mobile Device Setup* UI | Additional Settings [page 51] |

| Operation | Reference |
| --- | --- |
| Setting an expiration warning period for all user accounts | Additional Settings [page 51] |

## Required Role Assignments

Users can only access OTP tools if they have been assigned the corresponding roles. Provided you have permission to access SAP NetWeaver Administrator, you can assign OTP roles to the corresponding user management engine (UME) groups, roles, or users. For more information, see Assigning Principals to UME Roles or Groups.

You should assign the following OTP roles to the UME groups or users that use OTP authentication:

- `OTP_ADMINISTRATOR`
  This role is assigned to administrators to use *One-Time Password Administration* UI.
- `OTP_USER`
  This role is assigned to users to set up their authenticator apps for OTP authentication.
- `OTP_ONLINE_USER`
  This role is assigned to users for online account setup, which requires a confirmation code.

If you decide not to use the OTP roles mentioned above, you need to assign the following actions to the UME roles that you will use for OTP authentication:

- `OTP_ADMINISTRATION`
  This is an action for administrators using *One-Time Password Administration* UI.
- `OTP_ACTIVATION`
  This is an action for users setting up their authenticator apps.
- `OTP_ONLINE_ACTIVATION`
  This is an action for authenticator users for online account setup.

## Related Information

One-Time Password Authentication Installation and Upgrade Guide [page 4]
Configuring TOTPLoginModule and RBALoginModule [page 8]
Configuring Logon to Supported Systems [page 25]
Configuring Single-Factor Authentication [page 28]
Configuring Two-Factor Authentication [page 30]
Configuring an OTP-Related Logon Application [page 38]
Configuring the Online Account Setup [page 41]
Performing the Administrative Setup [page 42]
Managing User Accounts [page 43]
Using Mobile Single Sign-On [page 44]
Additional Settings [page 51]

One-Time Password Authentication Developer Guide [page 55]
One-Time Password Authentication User Guide [page 95]

## 1.2.1 Configuring TOTPLoginModule and RBALoginModule

SAP Single Sign-On provides two login modules which perform two-factor authentication. These are `TOTPLoginModule` and `RBALoginModule`. You can use the `TOTPLoginModule` when you want the first factor to be one of the login modules available on the AS Java, and the second factor to be a one-time password (passcode). You can use the `RBALoginModule` when you want to use the login modules available on the AS Java as first and second factor. This topic handles mainly the `TOTPLoginModule`, but most of these concepts are also valid for the `RBALoginModule`. You can find more examples in the Examples topic.

### How to Use the TOTPLoginModule

If you want to use the `TOTPLoginModule`, you need to add it to the authentication stack of your application. For more information, see Policy Configurations and Authentication Stacks. The `TOTPLoginModule` can use one or more login modules available in AS Java as the first authentication factor. For more information about the login modules, see Login Modules. You can specify the first factor login module or modules on the *One-Time Password Administration* UI at `http://<host>:<port>/ssoadmin/otp`, *Settings* tab, below the *Two-Factor Authentication*section.

### How the Authentication Works

1. When a user tries to open the protected application, the `TOTPLoginModule` is triggered as part of the corresponding authentication stack, which protects the application.
2. The `TOTPLoginModule` is now at the first stage and calls its first factor login module or modules, until a login module finds a user principal (for example username) that correspond to the authenticating user.
3. The `TOTPLoginModule` fails the authentication in order to trigger the second stage, where the user is requested to provide a passcode as second factor.
4. The user provides the passcode, the `TOTPLoginModule` succeeds and, depending on the rest of the login modules in the authentication stack and their flags, the overall authentication either succeeds or fails.

### Interaction Between the TOTPLoginModule and Other Login Modules

If you have an authentication stack with login modules, the `TOTPLoginModule` can replace this whole authentication stack (with some limitations), or it can replace a single login module from this authentication stack.

- Replacing the whole authentication stack

If you have configured an authentication stack with a few login modules, you can replace all these login modules only with one `TOTPLoginModule`. This `TOTPLoginModule` should have the same initial login modules configured as first factor login module. For more information on how to configure the first factor login module, see Configuring Two-Factor Authentication [page 30].
The login module options for the first factor login modules must be specified as `TOTPLoginModule` login module options. For more information, see the handling of login module options in the format *<login module>.<login module option>* in One-Time Password Login Module Options [page 13].

> ⚠ Caution
>
> When using only the `TOTPLoginModule` instead of a login module stack, keep in mind that the first factor login modules set to the `TOTPLoginModule` will be handled as with the *Sufficient* flag.

- Replacing one login module in the authentication stack
  If you want to enhance a single login module from your authentication stack with one-time password as a second factor, you need to replace this login module with the `TOTPLoginModule` and set the replaced login module as a first factor login module.

  > i Note
  >
  > If you are combining several login modules to create your authentication stacks, keep in mind that the order and flags of these login modules are very important for the authentication. If you use the `TOTPLoginModule`, there should be no other login modules with the *Requisite*, or *Required* flag after the `TOTPLoginModule`. For more information, see Policy Configurations and Authentication Stacks.

For more examples, see Examples [page 9].

## 1.2.1.1 Examples

### Example: 1. Setting Up Login Either with Password or with Passcode

John Miller is an administrator at MyCompany corporation. He has an application protected with user e-mail and password. He wants to configure the system to require authentication with a passcode in addition to the password.

John Miller has to do the following:

1. In SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa`, choose *Configuration* > *Authentication and Single Sign-On* > *Authentication* > *Components* and replace the `BasicPasswordLoginModule` with `TOTPLoginModule` in the authentication stack of the application.
2. Add a login module option for `TOTPLoginModule` with the name **`UserMappingMode`** and the value **`Email`**. This allows the customized logon application to correctly visualize the required user principal.
3. Add a login module option with name **`BasicPasswordLoginModule.UserMappingMode`** and the value **`Email`**. This instructs the `BasicPasswordLoginModule` to resolve the user by his/her e-mail.
4. On the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`, *Settings* tab, specify `BasicPasswordLoginModule` in the *First Factor Login Module* field.

5. Save the configurations.

Once this configuration has been completed, all employees of MyCompany will be able to log on to the system by entering an e-mail address and a password, or an e-mail address and a passcode generated by an authenticator mobile application.

## **Example:** 2. Setting Up Login with Certificate and Passcode

John Miller is an administrator at MyCompany corporation. He has an application protected by X.509 certificate authentication. He wants to add a passcode as a second factor. Assuming the CN field of the employees' certificates contains their corporate e-mail, and the application is already protected by `ClientCertLoginModule`, John Miller has to do the following:

1. In SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa`, choose *Configuration* > *Authentication and Single Sign-On* > *Authentication* > *Components* and replace the existing `ClientCertLoginModule` with `TOTPLoginModule`.
2. Add the login module option **`ClientCertLoginModule.Rule1.UserMappingMode`** with value **`Email`**.
3. Add the login module option **`ClientCertLoginModule.Rule1.AttributeName`** with value **`CN`**.
4. Add the login module option **`ClientCertLoginModule.Rule1.getUserFrom`** with value **`subjectName`**.
5. On the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`, *Settings* tab, specify `ClientCertLoginModile` in the field *First Factor Login Module*.
6. Save the configurations.

Once this configuration has been completed, all employees of MyCompany will be able to log on to the protected application by providing their certificate and by entering a passcode as well.

## **Example:** 3. Setting Up Login with Password, or Password and Passcode According to the Policy Script

John Miller is an administrator at MyCompany corporation. He has an application protected with username and password, and he wants to add a passcode as a second factor. The users in Germany should authenticate with a password only, while users outside of Germany should authenticate with a password and a passcode. John has to do the following for this configuration:

1. In SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa`, choose *Configuration* > *Authentication and Single Sign-On* > *Authentication* > *Components* and replace the `BasicPasswordLoginModule` with `TOTPLoginModule` in the authentication stack of the application.
2. On the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`, *Settings* tab, specify `BasicPasswordLoginModule` in the *First Factor Login Module* field.
3. Select the *Policy* checkbox in order to activate the policy.
4. Log on to the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts`.
5. Create a policy script of type *Procedure* that requires authentication according to the location of the user. You can find an example of a policy script in this step. For more information on how to create a policy script, see Working with Policy Scripts.

> **Example**
>
> ```
> function onFirstStageLogin(config, context, result) {
>   var loginInfo = context.getLoginInfo();
>   var user = loginInfo.getUser();
>   var userCountry = user.getCountry();
>
>   if ("DE".equalsIgnoreCase(userCountry)) {
>     result.doNotRequireSecondFactor();
>   }
> }
> ```

6. On the *One-Time Password Administration* UI, *Settings* tab, below the *Two-Factor Authentication* section, press the *Policy Script...* button and choose the policy script that was just created.
7. Save the configurations.

Once this configuration has been completed, all users from Germany will be able to log on with their logon ID and password, while all other users log on with basic credentials and a passcode.

## Example: 4. Protecting the Default *ticket* Authentication Template with Two-Factor Authentication Using One-Time Passwords

John Miller is an administrator at MyCompany corporation. He wants to configure the system to require authentication with two factors when accessing applications protected with the *ticket* authentication template.

John Miller has to do the following:

1. Log on to SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa` and navigate to *Configuration tab* > *Authentication and Single Sign-On* > *Authentication tab* > *Components*.
2. Filter for type **`Template`** and choose the policy configuration with name *ticket*.
3. Replace the `BasicPasswordLoginModule` with `TOTPLoginModule` in the authentication stack of the application.

> **Note**
>
> If there are login modules with the *Requisite* or *Required* flag below the `TOTPLoginModule` module, change these flags to *Sufficient*.

4. On the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`, *Settings* tab, specify `BasicPasswordLoginModule` in the *First Factor Login Module* field.
5. Save the configurations.

Once this configuration has been completed, all employees of MyCompany will be able to log on to any application protected with the *ticket* template (for example the portal page) with a one-time password, in addition to the user name and password, or SAP Logon Ticket.

## **Example:** 5. Using RBALoginModule to protect the Portal Page with BasicPasswordLoginModule as First Factor and SAML2LoginModule as Optional Second Factor

John Miller is an administrator at MyCompany corporation. He has a portal application protected by the default *ticket* authentication template, which authenticates users via SAP Logon Tickets or via basic authentication. For users matching certain criteria, John wants to configure the system to require SAML2 authentication after the successful basic authentication as first factor. To achieve this, John uses the risk based authentication offered by the `RBALoginModule`.

John Miller has to do the following:

1. Log on to SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa` and navigate to *Configuration tab* > *Authentication and Single Sign-On* > *Authentication tab* > *Components*.
2. Filter for type **`Template`** and choose the policy configuration with name *ticket*.
3. Make sure that the authentication stack has the following login modules and login module options:
   1. `EvaluateTicketLoginModule` with flag *Sufficient*
   2. `RBALoginModule` with flag *Requisite* and the following login module options:
      - **`tfa.first.factor.login.module`** with value **`BasicPasswordLoginModule`**
      - **`tfa.second.factor.login.module`** with value **`SAML2LoginModule`**
   3. `CreateTicketLoginModule` with flag *Optional*
4. Log on to the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts`.
5. Create a policy script of type *Procedure* that checks if SAML2 authentication is required depending on a user attribute defined in the policy script.
   Here is an example:

> **i** Note
>
> This policy script example is created for the *ticket* authentication template with authentication stack as mentioned in the steps above.

> **Example**

```
function disableMobileSSO(logger, loginInfo , config)
{
  var secondFactor;
  if (loginInfo)
  {
    secondFactor = "SAML2LoginModule";
  }
  else
  {
    secondFactor = "EvaluateTicketLoginModule";
  };
  config.setProperty("tfa.second.factor.login.module", secondFactor );
}
function onInitialize(config, context)
{
  var logger = context.getLogger();
  var loginInfo = context.getLoginInfo();
  disableMobileSSO(logger, loginInfo , config);
}
function onFirstStageLogin(config, context, result)
{
```

```
  var loginInfo = context.getLoginInfo();
  var user = loginInfo.getUser();
  config.setProperty("tfa.second.factor.login.module",
"SAML2LoginModule" );
  if (!requireSecondFactor(user))
  {
     result.doNotRequireSecondFactor();
  }
}
function requireSecondFactor(user)
{
  //specify below the user attribute which defines when and if the
SAML2LoginModule should be called
  var userCountry = user.getCountry();
  if (userCountry === 'DE')
  {
     return false;
  }
  else
  {
     return true;
  }
}
```

6. On the *One-Time Password Administration* UI, *Settings* tab, below the *Two-Factor Authentication* section, press the *Policy Script...* button and choose the policy script that was just created.
7. Save the configurations.
8. Select the *Policy* checkbox in order to activate the policy.

Once this configuration has been completed, all employees of MyCompany will be able to log on to any application protected with the *ticket* template (for example the portal page) with a `BasicPasswordLoginModule` as first factor and optionally with `SAML2LoginModule`. Before `SAML2` is required as second factor, the policy script verifies if the second factor is required for this specific user, depending on a user attribute defined in the policy script. If it is not required, the user is able to log in only with user name and password. If a second factor is required, the is `SAML2LoginModule` triggered.

> **!** Restriction
>
> Keep in mind that the mobile single sing-on, where both authentication credentials are received simultaneously, does not work with this setup.

## 1.2.1.2 One-Time Password Login Module Options

All of these options are optional. Some of them are configurable in the *One-Time Password Administration* UI. When you declare options for the `TOTPLoginModule` in SAP NetWeaver Administrator, you overwrite the central ones set in the *One-Time Password Administration* UI. This might be necessary when you set custom options for a specific policy configuration (application). If an option is not set explicitly as a login module option, `TOTPLoginModule` will use the default value.

TOTPLoginModule Options

| Name | Value | Description |
| --- | --- | --- |
| mode | *otp&pwd* | When the mode is set to *otp&pwd* (otp and pwd), two-factor authentication is required. You use two login modules to authenticate the user:<br>• `TOTPLoginModule` to authenticate the user with a user name and a passcode.<br>• Another login module to authenticate the user with credentials other than the passcode. For this reason, the `tfa.first.factor.login.module` option is taken into account.<br>**i Note**<br>If `mode` is not set explicitly in SAP NetWeaver Administrator, `TOTPLoginModule` will act like in *otp&pwd* mode. |
| | *otp* | If the mode is set to *otp*, single-factor authentication is required. You use the `TOTPLoginModule` to authenticate the user with a user name and a passcode. |

| Name | Value | Description |
| --- | --- | --- |
| | *otp\|pwd* | If the mode is set to *otp\|pwd* (otp or pwd), single-factor authentication is required. You can use either the `TOTPLoginModule` to authenticate the user with a user name and a passcode, or another login module to authenticate the user with credentials other than the passcode. In *otp\|pwd* mode, it is sufficient if one of the login modules succeeds.<br><br>For this mode you use `pwd.login.module.name` option for the non-OTP login module.<br><br>**i Note**<br>If the mode is set to *otp\|pwd*, basic authentication is taken into account for the non-OTP factor. |
| policy | <name of policy> | This option is used for two-factor authentication, and its value must match with the name of the policy script configured in the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts`. For more information, see Policy Scripts Implementation Guide<br><br>**i Note**<br>The policy is executed when the *tfa.policy.activated* option is enabled. |
| UserMappingMode | *LogonID* | Specifies the user mapping mode. This tells the login module how to retrieve the user. The mapping property for this value is the logon ID.<br><br>**i Note**<br>This is the default value. |
| | *Email* | The mapping property is the user's e-mail address. |

| Name | Value | Description |
| --- | --- | --- |
| | *VirtualUser* | The authenticated user is mapped to a virtual user. This means that if no such user exists in the UME database. Instead, the user is temporarily created for the current session. For more information, see Two-Factor Authentication with Virtual Users [page 36]. |
| tfa.enable.xsrf.protection | *yes*; *no* | Specifies if users receive a message when the system detects a potential XSRF attack. |
| log.http.headers | <string><br>**Note**<br>If you specify multiple headers, define them in a comma separated list. | Specifies what headers are used and shown in the logs.<br>One or a combination of the following headers is used by default:<br>*Host*, *Referer*, *User-agent*, *Accept*, *Accept-Language*, *Connection*, *Cookie* |
| The following options overwrite the settings in the *One-Time Password Administration* UI: | | |
| otp.allow.concatenated.password.and.passcode | *yes*; *no* | This option is not applicable for the OTP-related logon application. It specifies if a user can concatenate password and passcode in one field. For more information, see Configuring an OTP-Related Logon Application [page 38].<br>**Note**<br>The option is disabled by default. |
| otp.passcode.separator | <string> | Specifies the required separator if the password and passcode are entered in one field.<br>**Note**<br>This option is used if `otp.allow.concatenated.password.and.passcode` is enabled.. |
| otp.remember.client | *yes*; *no* | Specifies if a persistent cookie is issued for single-factor authentication. |

| Name | Value | Description |
| --- | --- | --- |
| otp.cookie.expiry | <integer between 1 and 365> | Validity in days of the persistent cookie used for single-factor authentication. |
| otp.cookie.http_only | *yes*; *no* | The cookie cannot be accessed from the script in the browser.<br>This option is used for single-factor authentication. |
| otp.cookie.secure | *yes*; *no* | The cookie is sent to the browser only if the HTTPS scheme is used for secure connections.<br>This option is used for single-factor authentication. |
| otp.login.consecutive.passcodes.enforced | *yes*; *no* | Specifies whether two passcodes are required for single-factor authentication. |
| otp.require.user.confirmation | *yes*; *no* | Specifies if user confirmation is required for automatic logon. |
| tfa.passcode.via.sms | *yes*; *no* | Specifies if the passcode is sent via SMS in two-factor authentication.<br>**i Note**<br>All SMS options are applicable only when this option is enabled. |
| sms.destination | <name of destination> | Configure this property if you set the passcode to be sent via SMS. For more information, see Configuring Authentication with a Random Passcode Sent by SMS, E-Mail, or Another Channel [page 32] . |
| sms.max.failed.attempts | <positive integer> | The maximum number of failed logon attempts with a passcode sent via SMS. |

| Name | Value | Description |
| --- | --- | --- |
| sms.message.text | <string> | Text for the SMS message that has to include the [passcode] placeholder.<br><br>**→ Recommendation**<br>Use Latin alphanumeric characters. Some mobile devices do not support all alphabets. |
| sms.quota | *yes*; *no* | Specifies if SMS quota is enabled. |
| sms.quota.overall | <positive integer> | Allowed SMS messages for all users for the last 24 hours.<br><br>Applicable if SMS quota is enabled. |
| sms.quota.per.user | <positive integer> | Allowed SMS messages per user for the last 24 hours.<br><br>Applicable if SMS quota is enabled. |
| sms.send.group | <string> | SMS messages are sent to the members of the specified group. |
| sms.token.length | <integer between 6 and 20> | Length of passcodes sent via SMS. |
| sms.token.validity | <positive integer> | Validity of passcodes sent via SMS. |
| sms.ume.attribute.for.number | <string> | UME attribute for a mobile number. Use this property to define from which UME attribute the application takes the mobile number for the SMS.<br><br>**i Note**<br>The default value is *mobile*. |

| Name | Value | Description |
| --- | --- | --- |
| tfa.first.factor.login.module | <login module>[,<optional login module 1>,<optional login module 2>, ...] | The `tfa.first.factor.login.module` option is used for *otp&pwd* and *otp* modes.<br><br>The value of this option is the display name of the login module, that authenticates the user with credentials other than the passcode. You can specify multiple login modules, by separating them by commas. If you specify multiple login modules, they will be evaluated in the sequence of their occurrence, until a rule succeeds. If none of the login modules succeeds, the first authentication factor fails.<br><br>**Example:**<br><br>Set the *tfa.first.factor.login.module* option to *SPNegoLoginModule,ClientCertLoginModule,BasicPasswordLoginModule*.<br><br>Depending on their working environment, the users authenticate as follows:<br><br>• When the users are working from the corporate network, they will log in with the *SPNegoLoginModule* login module as first factor.<br>• When the users are working from outside the corporate network (for example, at a customer visit), but on a corporate mobile device with X.509 certificate, they will authenticate via *ClientCertLoginModule* as first factor.<br>• When the users are at home, working on their private mobile devices, they will log in with their username and password as first factor.<br><br>You can set login modules available in the SAP NetWeaver Administrator at |

| Name | Value | Description |
| --- | --- | --- |
| | | `http(s)://<host>:<port>/nwa`, *Configuration* > *Authentication and Single Sign-On* > *Authentication* > *Login Modules*. For more information, see Login Modules.<br><br>Some of the supported login modules are as follows:<br><br>• *BasicPasswordLoginModule*<br>The value of this option defines the login module that authenticates the user with credentials other than the passcode. The `tfa.first.factor.login.module` option is used for `otp&pwd` and `otp` modes.<br>When you log on with a password, the value of the option has to be `BasicPasswordLoginModule`. For more information about basic authentication, see Basic Authentication (User ID and Password).<br>• *ClientCertLoginModule*<br>When you log on with a certificate, the value of the option has to be `ClientCertLoginModule`.<br>For more information about certificate authentication, see X.509 Client Certificates.<br>• *SPNegoLoginModule*<br>When you log on with a Kerberos token, the value of the option has to be `SPNegoLoginModule`.<br>For more information about Kerberos authentication, see Using Kerberos Authentication<br>• *EvaluateTicketLoginModule*<br>Use this login module for SAML 2.0 authentication. For more information about the login module, see Configuring the AS Java to Accept Logon Tickets.<br>You also need to grant a new keystore permission with a *SAML2* |

| Name | Value | Description |
| --- | --- | --- |
| | | keystore view for the relevant domain in SAP NetWeaver Administrator. For this setting, navigate to *Certificates and Keys: Key Storage* > *Key Storage tab* > *Security* > *Permissions per Domain tab* > and choose *Modify*. For more information, see Using the AS Java Key Storage.<br>• *SAML2LoginModule*<br>Performs user authentication using the SAML assertions. For more information about configuring the use of this login module, see Configuring AS Java as a Service Provider.<br>You also need to grant a new keystore permission with a *SAML2* keystore view for the relevant domain in SAP NetWeaver Administrator. For this setting, navigate to *Certificates and Keys: Key Storage* > *Key Storage tab* > *Security* > *Permissions per Domain tab* > and choose *Modify*. For more information, see Using the AS Java Key Storage.<br>• <name of third-party login module><br>You can use a third-party login module as a factor for OTP authentication. For more information, see Integrating Third-Party Login Modules |

| Name | Value | Description |
| --- | --- | --- |
| <login module>.<login module option> | <value of login module option> | Once you have set up the first factor login module or modules, you can set up the corresponding login module options.<br><br>For example, you can define the following first factor login modules and login module options:<br><br>• Set the `tfa.first.factor.login.module` option to `SPNegoLoginModule,ClientCertLoginModule,BasicPasswordLoginModule`.<br>• Set the `BasicPasswordLoginModule.UserMappingMode` option to `Email`.<br>• Set the `ClientCertLoginModule.Rule1.getUserFrom` option to `subjectName`.<br>• Set the `ClientCertLoginModule.Rule1.AttributeName` to `CN`.<br><br>For more information, see Using Rules for User Mapping in Basic Password Login Module, Using Rules for User Mapping in Client Certificate Login Module and Integrating Third-Party Login Modules. |
| tfa.remember.client | *yes*; *no* | Specifies if a persistent cookie is created for two-factor authentication. |
| tfa.cookie.expiry | <integer between 1 and 365> | Validity of persistent cookie for two-factor authentication. |
| tfa.cookie.http_only | *yes*; *no* | The cookie is not accessible from the script of the browser.<br><br>This option is used for two-factor authentication. |

| Name | Value | Description |
| --- | --- | --- |
| tfa.cookie.secure | *yes*; *no* | The cookie is sent to the browser only if the HTTPS scheme is used for secure connections.<br>This option is used for two-factor authentication. |
| tfa.accept.client.cookie | *yes*; *no* | This option is used for a policy script and specifies if an application can accept a persistent client cookie. |
| tfa.issue.client.cookie | *yes*; *no* | This option is used for a policy script and specifies if a persistent client cookie can be issued. |
| tfa.issue.client.cookie.require.consent | *yes*; *no* | Equivalent to the *Require User Consent* option in the *One-Time Password Administration* UI. Specifies if *Trust this computer* or *Trust this device* checkbox is available for two-factor authentication. If a user selects the checkbox, he or she will log on with a single factor next time. |
| tfa.policy.activated | *yes*; *no* | Specifies if the application can use a policy script.<br>**i Note**<br>This option is disabled by default. |
| otp.use.external.passcode.validation | *yes*; *no* | This option is enabled when users log on with passcodes generated by a third-party passcode provider. The option specifies if the application will validate such external passcodes.<br>**i Note**<br>This option is disabled by default. |

## Related Information

Configuring TOTPLoginModule and RBALoginModule [page 8]

### 1.2.1.3 Risk-Based Authentication Login Module Options

You need the `RBALoginModule` for two-factor authentication cases without OTP authentication, when a user authenticates with a client certificate and a password instead of a passcode for example.

RBALoginModule Options

| Name | Value | Description |
|---|---|---|
| `tfa.first.factor.login.module` or `tfa.second.factor.login.module` | *BasicPasswordLoginModule* | The value of the this option defines the first or the second factor of authentication. For more information about basic authentication, see Basic Authentication (User ID and Password) |
| | *ClientCertLoginModule* | When you log on with a certificate, the value of the option has to be `ClientCertLoginModule`. For more information about certificate authentication, see X.509 Client Certificates. |
| | *SPNegoLoginModule* | When you log on with a Kerberos token, the value of the option has to be `SPNegoLoginModule`. For more information about Kerberos authentication, see Using Kerberos Authentication |
| | <name of third-party login module> | You can use a third-party login module as a factor for OTP authentication. For more information, see Integrating Third-Party Login Modules |
| policy | <name of policy> | This option is used for **otp&pwd** mode only, and its value must match with the name of the policy script created in the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts`. For more information, see Working with Policy Scripts |

| Name | Value | Description |
| --- | --- | --- |
| <login module>.<login module option> | <value of login module option> | You can use all options from of the login modules used. You can define a user mapping for basic password login module for example with the option *BasicPasswordLoginModule.UserMappingMode*.<br><br>More Information:<br>• Using Rules for User Mapping in Basic Password Login Module<br>• Using Rules for User Mapping in Client Certificate Login Module<br>• Integrating Third-Party Login Modules |
| log.http.headers | <string><br><br>**i Note**<br>If you specify multiple headers, define them in a comma separated list. | Specifies which headers are used and shown in the logs.<br><br>One or a combination of the following headers is used by default:<br><br>*Host*, *Referer*, *User-agent*, *Accept*, *Accept-Language*, *Connection*, *Cookie* |

### Related Information

Configuring TOTPLoginModule and RBALoginModule [page 8]

## 1.2.2 Configuring Logon to Supported Systems

To log on to systems with one-time password (OTP) authentication, you can apply the `TOTPLoginModule` to a relevant policy configuration and specify the type of authentication in the Administration UI.You can use two types of authentication: single-factor and two-factor authentication.

### Prerequisites

You have set your policy configuration to use the `TOTPLoginModule` in SAP NetWeaver Administrator. For more information, see Configuring TOTPLoginModule and RBALoginModule [page 8].

## Related Information

Configuring One-Time Password Authentication for Secure Login [page 26]
Configuring One-Time Password Authentication for SAML 2.0 Identity Provider [page 27]
Configuring One-Time Password Authentication for Web Applications [page 28]

# 1.2.2.1 Configuring One-Time Password Authentication for Secure Login

## Context

To log on to a system using Secure Login Client, you can create an authentication profile that serves as a pointer to the configuration of the `TOTPLoginModule` and download the client authentication policies to the Secure Login Client.

## Procedure

1. Log on to Secure Login Administration Console (SLAC) at `http(s)://<host>:<port>/ssoadmin/sls`
2. To create an authentication profile, navigate to *Client Management* > *Authentication Profiles* and choose *Create*.

   The *Create New Authentication Profile* wizard appears. For more information about how to configure the new authentication profile, see Related Information.
3. To create a profile group, navigate to *Client Management* > *Profile Groups* and choose *Create*.

   A *New Profile Group* pop-up window appears. For more information about how to enter the group name and set the proper properties, see Related Information.
4. To add the authentication profile to the profile group, go to the *Details of <name of profile group>* pane and choose *Edit*.
5. Go to the *Profile group* tab and choose *Add*.
6. To download the client authentication policies of the Secure Login Server to the Secure Login Client in a profile group, navigate to *Client Management* > *Profile Groups* > *Download Policy*. For more information about how to download client policies files, see Related Information.

## Related Information

Creating an Authentication Profile Pointing to a Policy Configuration
Creating a Profile Group of Client Authentication Profiles

# 1.2.2.2 Configuring One-Time Password Authentication for SAML 2.0 Identity Provider

## Prerequisites

You have configured your identity provider with the relevant trust and identity federation settings. For more information, see Related Information.

## Context

To log on to a system using the AS Java Identity Provider, you can create a custom authentication context and map it to the `TOTPLoginModule` that will be used by the identity provider to authenticate users.

## Procedure

1. Log on to SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa`.
2. To create a custom authentication context, navigate to *Authentication and Single Sign-On: SAML 2.0* > *SAML 2.0* > *Local Provider* > *Authentication Contexts* tab and choose *Edit*.
3. Choose *Add*
4. In the *Create Authentication Context* window, enter the *Alias* and the *Name*, and choose *OK*.
5. Filter your authentication context and set the type as *Interactive*. The context should support *HTTP* and *HTTPS*. For more information, see Related Information.
6. To configure your identity provider to use your authentication context, navigate to *Identity Provider Settings tab* > *Supported Authentication Contexts pane*, and choose *Edit*.
7. Choose *Add*.
8. Select your authentication context and map it to the `TOTPLoginModule` in the *Authentication Context and Login Module Mapping* pop-up window.
9. Filter your authentication context in the *Supported Authentication Contexts* pane and choose *Copy to*.
10. Choose *Default HTTPS Authentication Contexts* and *Default HTTP Authentication Contexts*.

    The *Default HTTPS Authentication Contexts* and the *Default HTTP Authentication Contexts* appear in separate panes.
11. Save your configuration.

## Related Information

Configuring the Identity Provider
Optional Configurations

## 1.2.2.3 Configuring One-Time Password Authentication for Web Applications

### Context

To log on to a web application running on SAP NetWeaver AS for Java, you can apply the `TOTPLoginModule` to the policy configuration of your web application.

### Procedure

1. Log on to SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa`.
2. Navigate to *Authentication and Single Sign-On* > *Authentication tab* > *Components* and set the type to *Web*.
3. Select the web application you want to configure for logon with one-time password authentication.
4. To apply the `TOTPLoginModule` to the policy configuration of your web application, choose *Edit* and navigate to *Details of policy configuration pane* > *Authentication Stack tab*.
5. Choose *Add*.
6. From the dropdown list, select the `TOTPLoginModule`.
7. Set the processing flag for your login module. For more information about login module flags, see Related Information.

### Related Information

Policy Configurations and Authentication Stacks
Configuring Logon to Supported Systems [page 25]

## 1.2.3 Configuring Single-Factor Authentication

You can use single-factor authentication if users log on with only one factor: a passcode, a password, an X.509 certificate, or other. The passcode has to be generated by a mobile device with an installed authenticator mobile application.

## Prerequisites

You have configured the `TOTPLoginModule` to support `otp` or `otp|pwd` modes.

For more information, see Configuring TOTPLoginModule and RBALoginModule [page 8].

## Context

You can do the following settings for the single-factor authentication:

- You can require confirmation from users when they automatically log on to an application.
  This might be necessary for protection against logon XSRF attacks in mobile single sign-on scenarios when users can log on to applications without providing a passcode.
- You can require two distinct passcodes from users before they log on to an application.
  This might be necessary for security reasons if using only one passcode might result in security being compromised.
- You can set the system to create a persistent cookie when the user initially provides two distinct passcodes.
  You can set the following for this cookie:
  - Validity
    The cookie is valid until it expires or is revoked. You revoke such a cookie for specific users under the *Users* tab of the *One-Time Password Administration* UI. For more information about the management of user accounts, see Related Information.

    > **i Note**
    >
    > The default value is 30 days.

  - HTTP only
    This property shows that the persistent cookie is not accessible from the script of the browser.

    > **i Note**
    >
    > This property is enabled by default.

  - Secure
    The persistent cookie is sent to the browser only if the HTTPS scheme is used for secure connections.

    > **i Note**
    >
    > This property is enabled by default.

## Procedure

1. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
2. Choose the *Settings* tab.

3. Choose the *Edit* button.
4. Under the *Single-Factor Authentication* section, select the required configurations.
5. Save your configuration.

### Related Information

Using Mobile Single Sign-On [page 44]
Managing User Accounts [page 43]

## 1.2.4 Configuring Two-Factor Authentication

You need to configure two-factor authentication if you would like users to provider two independent factors for strong authentication.

### Overview

When you enable two-factor authentication, users log on with credentials defined by the first factor login module and usually a passcode as a second factor. The first factor login module can be defined in the following ways:

- In the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`
  The first factor login module that you configure in the *One-Time Password Administration* UI is used centrally for all applications. Furthermore, you can specify a list of login modules and then the first matching login module from the list will be used. Here you can use any login modules that are supported by SAP NetWeaver Administrator.
- In the SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa`
  You can set the first factor login module as a login module option of `TOTPLoginModule` in order to overwrite the central configuration. This is necessary for example if you would like to configure an application that does not use the central settings. For more information about the login module configuration, see Related Information.

You can also set the following additional options for two-factor authentication:

- You can allow the creation of a persistent cookie when a user initially enters both factors.
  The cookie is used as the first authentication factor. The user then logs on with the second factor (passcode, password, or another) until the cookie expires or is revoked.
  You can set the following for this cookie:
  - Require User Consent
    The cookie is issued only when the user selects *Trust this computer* or *Trust this device* checkbox on the logon page.
  - Validity
    The cookie is valid until it expires or is revoked. You revoke such a cookie for specific users under the *Users* tab of the *One-Time Password Administration* UI. For more information about the management of user accounts, see Related Information.

> **Note**
>
> The default value is 30 days.

- HTTP only
  This property shows that the persistent cookie is not accessible from the script of the browser.

  > **Note**
  >
  > This property is enabled by default.

- Secure
  The persistent cookie is sent to the browser only if the HTTPS scheme is used for secure connections.

  > **Note**
  >
  > This property is enabled by default.

- You can configure the system to send the passcode via SMS.
  You can do this if users do not have supported devices that allow the installation of an authenticator mobile application. , you need to develop a policy script.
- You can enable the use of a policy script.
  The policy scripts can be used for the following cases:
  - Controlling conditions for two-factor authentication
    You can develop a script setting conditions for a single or two-factor authentication in accordance with the location that a user is logging on from for example.
  - Sending passcodes via email
    You can develop a script for the passcodes to be send via out-of-band methods.
  - Validating passcodes generated by a third-party passcode provider.
    You can develop a script to validated passcodes generated by a third-party provider and to show the result of the user authentication.

  For details how to develop policy scripts, see Related Information.

## Related Information


Configuring TOTPLoginModule and RBALoginModule [page 8]
Configuring Logon to Supported Systems [page 25]
Configuring Authentication with a Passcode Generated by an Authenticator App [page 32]
Configuring Authentication with a Random Passcode Sent by SMS, E-Mail, or Another Channel [page 32]
Configuring External Passcode Validation [page 35]
Two-Factor Authentication with Virtual Users [page 36]
One-Time Password Authentication Developer Guide [page 55]

### 1.2.4.1 Configuring Authentication with a Passcode Generated by an Authenticator App

This is a configuration for users that have an authenticator mobile application installed on their mobile devices.

#### Prerequisites

The application is configured to use two-factor authentication, which is the default setting. For more information, see Configuring TOTPLoginModule and RBALoginModule [page 8].

#### Procedure

1. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
2. Choose the *Settings* tab.
3. Choose the *Edit* button.
4. Under the *Two-Factor Authentication* section, specify the first factor login module and whether a persistent cookie is issued.
5. To use a policy script, choose the *Policy Script...* button, enter the script in the new pop-up window, and select the *Policy* checkbox.

   For information about how to develop policy scripts, see Related Information.
6. Save your configuration.

#### Related Information

One-Time Password Authentication Developer Guide [page 55]
Configuring Two-Factor Authentication [page 30]

### 1.2.4.2 Configuring Authentication with a Random Passcode Sent by SMS, E-Mail, or Another Channel

For users who cannot install an authenticator application on their mobile devices, you can configure the passcode to be sent by SMS.

> **i Note**
>
> The SMS configuration works out-of-the-box with an SAP Messaging Service, which you need a separate contract for. To contact the SAP Messaging Service team, use *sapmobileservices@sap.com*. If you would

like to use a third-party Short Message Service (SMS) Gateway, you have to write a policy script that implements the API of the third-party SMS Gateway. For an example script, see SAP Note 2225027.

You can also write a policy script so that the passcode is sent by e-mail. For more information about the out-of-band methods, see Related Information..

## Prerequisites

The application is configured to use two-factor authentication, which is the default setting. For more information, see Configuring TOTPLoginModule and RBALoginModule [page 8].

## Procedure

1. Log on to SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa`.
2. Configure system VM parameters.

   For more information about the Java system properties, see Related Information.
3. Configure the SMS destination.

   For more information about the HTTP destination, see Related Information.
4. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
5. Choose the *Edit* button under the *Settings* tab.
6. Under the *Two-Factor Authentication* section, specify the first factor login module, whether the persistent cookie will be issued, and the cookie's options.
7. Select the *Send passcode by SMS* checkbox.
8. To use a policy script, choose the *Policy Script...* button, enter the script in the new pop-up window, and select the *Policy* checkbox.

   For information about how to develop policy scripts, see Related Information.
9. Under the *SMS Gateway* section, specify the destination, group and UME attribute for mobile number configured in SAP NetWeaver Administrator.

   > **i Note**
   >
   > The group and the phone number are configured under the *Identity Management* section..

   You also have to define a message text that includes the *[passcode]* placeholder.

   > **→ Recommendation**
   >
   > For the SMS message, you should use Latin alphanumeric characters. Some mobile devices do not support characters from other alphabets.
10. Save your configuration.

## Example

As an administrator at Company A, Donna Moore would like to configure the system to send passcodes to partner users who cannot install an authenticator application and cannot set up their mobile devices. To do this, she proceeds as follows:

1. She logs on to SAP NetWeaver Administrator and goes to *Java System Properties: Overview*, chooses the *System VM Parameters* tab and configures the following properties for the selected SMS template:

| Name | Custom Calculated Value |
|---|---|
| http.nonProxyHosts | localhost\|*.companya.corp |
| http.proxyHost | proxy |
| http.proxyPort | 8080 |
| https.nonProxyHosts | localhost\|*.companya.corp |
| https.proxyHost | proxy |
| https.proxyPort | 8080 |

2. Donna goes to the *Destinations: Destinations* section and creates a new HTTP destination with name `SMS_GATEWAY` as she sets the destination URL, username, and password.
3. She sets a phone number for user Michael Adams with logon ID *m_adams* under *Identity Management: Overview* > *m_adams* > *Contact Information* by choosing the *Modify* button and entering the number in the mobile field.
4. She assigns this user to the *Partners* group under the *Assigned Groups* tab.
5. Donna goes to *Authentication and Single Sign-On: Authentication* and sets the following `TOTPLoginModule` option for her policy configuration (application):

| Name | Value |
|---|---|
| tfa.first.factor.login.module | BasicPasswordLoginModule |

6. She logs on to the *One-Time Password Administration* UI, selects the *Send passcode by SMS* checkbox, and enters values in the following fields:

| Field Label | Field Value |
|---|---|
| Destination Name: | SMS_GATEWAY |
| Send SMS to Members of Group: | Partners |
| UME Attribute for Mobile Number: | mobile |
| Message Template: | Please log on with the following passcode: [passcode]. |

After completing the configuration, Donna informs Michael that he can log on with passcodes sent by SMS. When Michael accesses the logon page, he first enters his username and password. After successful authentication with the password, a new page appears prompting Michael to enter the passcode he has just been sent by SMS. Michael retrieves the passcode from his phone and logs on to the application.

### Related Information

Java System Properties
Maintaining HTTP Destinations
Develop a Script for Risk-Based Authentication [page 83]
One-Time Password Authentication Developer Guide [page 55]
Configuring TOTPLoginModule and RBALoginModule [page 8]
Identity Management
Configuring Two-Factor Authentication [page 30]
Script 3: Set Conditions for Out-of-Band Methods [page 86]
One-Time Password Login Module Options [page 13]

## 1.2.4.3 Configuring External Passcode Validation

Users can log on with external passcodes generated by a third-party passcode provider such as RSA SecureID passcode.

To allow this kind of authentication, you need to make sure that the following prerequisites are met:

- The system that generates external passcodes is configured properly.
- The developers have implemented a policy script containing the `validatePasscode(config, context, result, username, passcode)` function.
  The function has to use the `validatePasscode(...)` method, and the policy script has to define when the validation is successful and when the user authentication fails. For more details how to develop this kind of script and what methods you can use, see Related Information.
- The `otp.use.external.passcode.validation` property is enabled for `TOTPLoginModule`.
  You can set this property in SAP NetWeaver Administrator as a login module option or in the policy script. For more details about how to set options for `TOTPLoginModule`, see Related Information.

### Related Information

Develop a Script for External Passcode Validation [page 90]
Policy Script Functions and Methods [page 57]
One-Time Password Login Module Options [page 13]

## 1.2.4.4 Two-Factor Authentication with Virtual Users

In SAP Single Sing-On version 3.0 and higher, two-factor authentication can work with virtual users. If a user passes the first factor authentication against an external data source (for example LDAP), but does not exist in the UME database, a temporary virtual user is created for the duration of the application session in the following cases:

- The user principal from the first factor login module is *VirtualUserPrincipal*. A new virtual user is then created and receives assignments for groups, roles and attributes as defined for the *VirtualUserPrincipal*.
- The value of the login module option *UserMappingMode* is *VirtualUser*. A new virtual user is then created with no group, role or attribute assignments.

> **i** Note
>
> The virtual user exists for the duration of the application sessions associated with that user (for example until the user logs out, or the session expires). If the user cancels the process after the first step, the virtual user exists until the session expires.

### 1.2.4.4.1 Examples of Two-Factor Authentication with Virtual Users

#### Example 1: Two-Factor Authentication with External Passcode Validation

**Prerequisites**

1. In the SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa`, you have protected an application with the login module `TOTPLoginModule`, and the following login module options:

| Name of the Login Module Option | Value of the Login Module Option | What Does It Do |
|---|---|---|
| `tfa.first.factor.login.module` | *SecureLoginModule20LDAP* | Verifies user's credentials against pre-configured LDAP server. |
| `UserMappingMode` | *VirtualUser* | If the user authenticated by the first factor login module does not exist in the UME database, a virtual user is created for the duration of the application session. |

2. You have created a policy for external passcode validation. For more information, see Develop a Script for External Passcode Validation [page 90].
3. You have configured external passcode validation. For more information, see Configuring External Passcode Validation [page 35].

### How Does It Work

Michael Adams is an employee of company A. Michael logs into an application protected with the login module `TOTPLoginModule`. The first factor login module is `SecureLoginModule20LDAP`, so Michael is requested to provide his LDAP credentials. Michael provides the credentials, and the first step of the two-factor authentication succeeds. As there is no user called Michael Adams in the UME data source, a new virtual user is created with no attribute, role, or group assignments. Michael is prompted to enter a passcode. After providing the passcode, this is validated externally, and Michael is successfully authenticated.

When Michael logs out of the application, or the application session expires, the virtual user is deleted.

## Example 2: Two-Factor Authentication with SAMLLoginModule and Passcode Sent via Email

### Prerequisites

1. You have configured SAP NetWeaver Application Server (AS) Java as a service provider. For more information, see Configuring AS Java as a Service Provider.
2. You have configured trust between the service provider and an identity provider. For more information, see Trusting an Identity Provider.
3. You have configured federation type virtual users, and have mapped SAML 2.0 attribute *email*. For more information, see Configuring Federation Type Virtual Users.
4. In the SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa`, you have protected an application with the login module `TOTPLoginModule`, and the following login module options:

| Name of the Login Module Option | Value of the Login Module Option | What Does It Do |
|---|---|---|
| `tfa.first.factor.login.module` | *SAML2LoginModule* | Uses SAML 2.0 as first factor authentication. |

5. You have created a policy to send an email with the passcode. For more information, see Configuring Authentication with a Random Passcode Sent by SMS, E-Mail, or Another Channel [page 32].

### How Does It Work

Michael Adams is an employee of company A. Michael opens the application protected with the login module `TOTPLoginModule` and is redirected to the identity provider in order to authenticate. After Michael authenticates on the identity provider, he is redirected back to the service provider. The first factor authentication passes successfully, and *SAML2LoginModule* creates *VirtualUserPrincipal*.

The user Michael Adams does not exist in the UME on the service provider, and does not have one-time password activation, so he could not provide a second factor. As the user principal created by the first factor login module is *VirtualUserPrincipal*, a new virtual user is created with groups, roles and attributes assigned as defined in *VirtualUserPrincipal*. An e-mail with the passcode is sent to Michael's e-mail address. Michael uses this passcode for the second factor authentication, and the overall authentication to the protected application succeeds.

### Example 3: Two-Factor Authentication with 3rd Party Login Module and Passcode Sent via Email

This example is very similar to Example 1, but instead of using the login module option `UserMappingMode`, a *VirtualUserPrincipal* is created via a policy script. In the policy script, you can specify attributes, groups, and roles to be assigned to the virtual user.

**Prerequisites**

1. In the SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa`, you have protected an application with the login module `TOTPLoginModule`, and the following login module option:

| Name of the Login Module Option | Value of the Login Module Option | What Does It Do |
|---|---|---|
| `tfa.first.factor.login.module` | <3rd party login module name> | Authenticates the user according to the 3rd party login module's logic. |

2. You have created a policy script in the *Policy Script Administration Console*, which replaces the user principal from the first step of authentication in the `onUserIdentified()` function with a new *VirtualUserPrincipal*, specifying details for the user (for example, email attribute). The policy also must send an e-mail to the user's e-mail address with the passcode for the second step of authentication. For more information on how to create new virtual principal, see Policy Script Functions and Methods [page 57].
3. You have set this policy in the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.

**How Does It Work**

Michael Adams is an employee of company A. Michael does not have a persistent user in the UME database, but the 3rd party login module can authenticate Michael against an external data source. Michael logs into an application protected with the login module `TOTPLoginModule`. The 3rd party login module authenticates Michael successfully. The policy script replaces the 3rd party user principal created by the 3rd party login module with a new `VirtualUserPrincipal`. In the policy script, you can specify roles, groups and attributes for the new virtual user that will be created. For this example, the e-mail attribute must be specified, so that an e-mail containing the passcode for the second step of authentication can be sent to the user. The virtual user is created successfully after the first step of the authentication, and an e-mail with the passcode is sent to Michael's e-mail address. Michael uses this passcode for the second factor, and the overall authentication to the protected application succeeds.

## 1.2.5 Configuring an OTP-Related Logon Application

For the proper one-time password (OTP) authentication flow with a browser client, you can configure a custom logon application that uses two steps of the logon process. This OTP-related logon application displays one

screen for the first factor and after successful validation of the first credential requires the credential for the second factor.

## Context

By setting the OTP-related logon application, you allow users to see the following improvements on the logon pages to any application:

- A QR code
  Users can scan the QR code to access the application or to add it as a favorite application for mobile single sign-on (SSO).
- Separate pages for password and passcode
  Users enter password and passcode on separate logon screens, but not concatenated.
- *Log on with SAP Authenticator* link
  Users can choose the *Log on with SAP Authenticator* link when they log on to an application from their mobile devices. Provided the application is added to the SAP Authenticator account, they log on with mobile SSO.
- Policy scripts
  The policy scripts developed to show messages on logon pages only work with the OTP-related logon application. That means if you use scopes in your policy script for full access or limited access authorizations, the logon screen works only with the OTP-related logon application.
- *Require user confirmation for automatic logon* setting in the *One-Time Password Administration* UI.
  Users will only see a confirmation screen during automatic logon if the OTP-related logon application is configured. This confirmation serves to protect the application from various attacks.
- *Trust this computer* or *Trust this device* checkbox
  Users can trust their computers or mobile devices during two-factor authentication by selecting a checkbox when the OTP-related logon application is configured. After selecting the checkbox, they will log on to the application with one factor only next time.
  You can perform this configuration in the *One-Time Password Administration* UI by selecting the *Require User Consent* chekbox under *Settings* > *Two-Factor Authentication* or by writing a policy script. For more details how to develop a policy script, see Related Information.

## Procedure

1. Log on to SAP NetWeaver Administrator at `http(s)://<host>:<port>/nwa`.
2. Choose the *Configuration* tab.
3. Choose the *Authentication and Single Sign-On* link.
4. Choose the *Authentication* tab and the *Properties* link.
5. Choose the *Modify* button.
6. Under the *Logon Application* section, set the *Alias of Application for Customizing Login Pages (ume.logon.application.ui_resources_alias)* property with value */otp_logon_ui_resources*.
7. Save your configuration.

## Related Information

Using Mobile Single Sign-On [page 44]
SAP Authenticator Mobile Application [page 95]
One-Time Password Authentication Developer Guide [page 55]

# Using a Company-Specific Logon Application

To use your own logon application with OTP authentication, you have to customize the OTP-related logon application instead of the default logon UI application. You can find the OTP-related logon application under the following path: <ASJava_Installation>`/j2ee/cluster/apps/sap.com/sso~otp~ear/servlet_jsp/otp_logon_ui_resources/sap.com~sso~otp~logon~ui.war`

When customizing your OTP logon application, note that you have to update the corresponding *.jsp* file, depending on how you access the application.

For example:

- For HTTP access, update the *logonPage.jsp*
- For HTTPS access, update the *certLogonPage.jsp*
- For access from mobile devices, update *certLogonPageMobile.jsp* or *logonPageMobile.jsp* in the folder *mobile*

For more information how to develop a custom logon screen, see Related Information.

> **i** Note
>
> In the `<project_name>/EARContent/META-INF/application-j2ee-engine.xml` file, change the reference to be to the OTP-related logon application (*sso~otp~ear*).
>
> **Sample Code**
>
> ```
> <?xml version="1.0" encoding="UTF-8"?>
> <!DOCTYPE application-j2ee-engine SYSTEM 'application-j2ee-engine.dtd'>
> <application-j2ee-engine
>     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
>     xsi:noNamespaceSchemaLocation="application-j2ee-engine.xsd">
>     <reference reference-type="hard">
>         <reference-target
>          provider-name="sap.com"
>          target-type="application">sso~otp~ear</reference-target>
>     </reference>
>     <provider-name>example.com</provider-name>
>     <start-up mode="manual"/>
> </application-j2ee-engine>
> ```

## Related Information

Developing a Custom Logon Screen

## 1.2.6 Configuring the Online Account Setup

Users who have *SAP Authenticator* 1.2.0 or higher installed on their mobile devices can set up their accounts by also providing a confirmation code.

### Context

The online account setup has the following specifics:

- An administrator has to assign the corresponding role or action to the users.
  Users can perform online account setup only if they have the necessary role or action assigned in the user management engine. For more details, see Related Information.
- Users receive the secret key and applications for their accounts.
  An administrator adds all user applications, and the users receive them with the online account setup of each device.
- Users have to enter a confirmation code to complete the setup.
  Administrators configure whether this code is received by e-mail or is shown in the Mobile Device Setup UI. Administrators can also set the validity of the confirmation code in minutes, the number of characters it contains, the mail server it is sent from, and the e-mail that it is sent with.
- An administrator can set up accounts on behalf of users.
  The administrator can perform the initial account setup on the user devices. Users are supposed to complete this simply by entering a confirmation code. For more details about the administrative setup, see Related Information.
- Users might be asked to configure a password.
  When administrators configure the password as mandatory, users will be requested to set a password. Otherwise, the account setup will be canceled.
- Users scan a QR code that comprises a setup URL or provide the URL manually.
  This URL contains the user data for the online account setup and has to be valid for the setup process to succeed.
- Users can update their accounts in SAP Authenticator.
  Users can update the applications for an account by choosing the *Update* button in SAP Authenticator. An administrator can change the applications for an account for example, and users will receive these changes via an update.
- Administrators can specify the size of the key used to encrypt the account setup data.
  This account data includes the secret key and applications.

> **i Note**
>
> Sometimes users might not be able to use an encryption key larger than 128 bits. This is due to the limited strength of Java Cryptography Extension (JCE). For a solution to this problem, see SAP Note 1240081. For more information about JCE, see Updating SAP JVM.

For configuration of the online account setup, proceed as follows:

### Procedure

1. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
2. Choose the *Devices* tab.
3. Choose the *Edit* button.
4. Configure the properties and add the applications.
5. Save your settings.

### Related Information

One-Time Password Authentication Administration Guide [page 5]
Performing the Administrative Setup [page 42]

## 1.2.6.1 Performing the Administrative Setup

An administrator can set up an account on behalf of a user.
The user then completes the account setup by providing the confirmation code received by e-mail.

> **i** Note
>
> This procedure is applicable for all OTP users. For more information about the roles or actions that these users can have, see Related Information.

### Prerequisites

The user's e-mail address is configured in the user management engine. For more information, see User Management of the Application Server Java

### Procedure

1. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
2. Choose the *Devices* tab.
3. Choose the *Set Up Device* button.

4. Search for a user and select it in the table in the wizard.
5. Complete the wizard and set up the account by scanning the QR code from the user's device.
6. Provide the device to the user and inform him or her that the confirmation code has been sent by e-mail.

### Related Information

One-Time Password Authentication Administration Guide [page 5]

## 1.2.7 Managing User Accounts

You can manage user accounts to resolve issues such as locked accounts, lost or stolen mobile devices, and possible logon XSRF attacks.

### Context

You perform actions of this kind for users when they have installed an authenticator application on their mobile devices and have set up their devices in the *Mobile Device Setup* user interface (UI). The authenticator is a mobile application that generates passcodes for systems that require one-time password authentication.

You can perform the following user management actions:

- Advanced user searches
  You can filter your user search by passcode length, digest algorithm used for passcode generation, user account expiration (with the *Expires in* field), and user account status (with the *OTP Status* field). If you search by *OTP Status*, you have the following options: *Enabled* (search for users with active accounts); *Expired* (search for users with expired accounts); *Expires soon* (search for users whose accounts are within the expiration warning period defined under the *Settings* tab); *Locked* (search for users whose accounts are locked); and *Not Confirmed* (search for users who have not yet provided the confirmation code and have not canceled the online account setup).
- Unlocking user accounts
  You can unlock user accounts if the users need the account before the automatic unlock time has passed. For more information about the additional settings, see Related Information.
- Disabling users
  You can disable user accounts, and then users are supposed to remove those accounts from their mobile devices. Users can also disable their accounts in the *Mobile Device Setup* UI if they have more than one registered device.
- Removing remembered clients (persistent cookies)
  If you have set this option for single-factor or two-factor authentication under the *Settings* tab of the *One-Time Password Administration* UI, you can revoke this cookie. You revoke the cookies by choosing the *Unregister Clients* button for the selected users.
- Setting a new validity for user accounts

You can set a new validity period for user accounts or their secret keys (a user account is active as long as its secret key is valid). The default validity is specified under the *Settings* tab. For more information about the additional settings, see Related Information.

### Procedure

1. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
2. Choose the *Users* tab.
3. Search for the users to manage.
4. Select these users.
5. Choose the button that corresponds to your operation.

   Once the operation has completed, you receive a success message.

### Related Information

Additional Settings [page 51]
Configuring Single-Factor Authentication [page 28]
Configuring Two-Factor Authentication [page 30]

## 1.2.8 Using Mobile Single Sign-On

If SAP Authenticator is installed on your mobile devices, you can use mobile single sign-on (SSO), based on one-time password (OTP) authentication. This feature allows you to save applications, which you can access next time without additional authentication (through SSO). You have to configure these applications with *TOTPLoginModule*.

As administrator, you should inform users how they can access these applications. You also have to comply with the URL requirements for these applications. When SAP Authenticator calls the URL, it replaces the `[username]` part of the URL with the specified account name and the `[passcode]` part with the generated passcode.

### Users Adding Applications

> **i Note**
>
> This operation is not applicable for accounts with online setup because applications of these accounts cannot be maintained by users.

Users can add applications to an SAP Authenticator account by scanning a QR code or by typing the application URL. The URL must have the following format:

`http(s)://<host>:<port>/<path>?j_username=[username]&j_passcode=[passcode].`

> ### Example
>
> Donna Moore is an administrator at company A. She would like to inform the users of her two applications, an SAP Enterprise Portal application and an SAP Fiori application, how to add the applications in SAP Authenticator for mobile SSO. Because she has configured the OTP-related logon application and has enabled *Remember client (persistent cookie)* under *Settings* in the *One-Time Password Administration* UI, the QR code is displayed on the logon page. She informs her users that they can scan the QR code on the logon page.
>
> Julie is a user who would like to add these applications in her SAP Authenticator mobile application. Julie adds both Web applications by scanning the QR codes on the applications' logon pages. After doing this, Julie logs on to both applications from SAP Authenticator without entering her username and passcode.

## Administrator Setup for SAP Authenticator Users

On the *Devices* tab in the *One-Time Password Administration* UI, below the *SAP Authenticator Configuration* section, administrators can add application bookmarks, or enable short-lived certificate provisioning for the SAP Authenticator users. They can also define whether a password is mandatory for the usage of the SAP Authenticator application. Users receive these applications and certificates in SAP Authenticator by going through the online account setup, or once done, by pressing the *Update* button in the SAP Authenticator application.

As an administrator, you can set up the following options for the SAP Authenticator users:

- *SAP Authenticator Password*
  Set the password for the SAP Authenticator users to be either mandatory or optional. If it is mandatory, the online account setup cannot be completed until the user sets a password for SAP Authenticator.
- *X.509 Certificate Provisioning*
  Enable or disable the provisioning of short-lived certificates to the SAP Authenticator users. On the mobile device, the certificates are stored in a shared keychain.

  > ### Note
  >
  > Certificate provisioning is available only for iOS mobile devices, and can be used only for SAP signed applications.

- *Set Login Authentication Profile*
  Choose from the available authentication profiles, defined in the *Secure Login Administration Console* (available at `http(s://<host>:<port>/ssoadmin/sls`), *Profile Management* > *Authentication Profiles*. You can view the certificate details on the *Certificate Configuration* tab below the selected authentication profile.

> **i Note**
>
> For security reasons, we recommend using short-lived certificates and updating them regularly. Since certificate revocation is not supported, we strongly advise against using long-lived certificates, although this is technically possible.

> **⚠ Caution**
>
> When configuring the profile for issuing X.509 certificates in the *Secure Login Administration Console*, you can only use the **(AUTH:USERID)** variable. The other variables are not supported by SAP Authenticator. If you set any of them, the certificate issuing process will fail.

- *Group for Certificate Provisioning*
  Specify to the members of which groups the certificate should be provisioned. The groups that administrators can add with the *Edit Groups* button are the ones in the user management engine (UME). If no group is specified, all OTP users receive the certificate.
- *Applications*
  Administrators can add or remove applications for the SAP Authenticator users. Administrators can also specify members of which groups can access each application. The groups that administrators can add with the *Edit Groups* button are the ones in the user management engine (UME). If no group is specified, all OTP users can log on to the application. SAP Authenticator users receive the changed content during account setup, or with each account update. Online activated accounts cannot be changed on mobile devices manually by the user.
  Administrators can also add an application that requires the users to authenticate with their corporate cards. This is valid only for mobile Single Sign-On scenarios (accessing applications from mobile devices). If you use shared devices (one mobile device is used by more than one user) you can set up the application to require an NFC card to authenticate every individual user.
  To set up this kind of application, proceed as follows:
  - Make sure that the URL of the application matches the following URL format: `http(s)://<host>:<port>/<path>?j_username=[username]&j_passcode=[passcode]&nfctagid=[nfc.tag.id]&signature=[signature]`
  - Configure a service user to be used by multiple users on the shared mobile devices. For more information, see Performing the Administrative Setup [page 42].
  - In the UME, add **`tagid`** as custom attribute of the user profile to the field *Administrator-Managed Custom Attributes*. For more information, see Adding Custom Attributes to the User Profile.
  - Create a policy script to match the tagID of the card with the authenticating user, and select this policy script on the *Settings* tab of the *One Time Password Administration* UI. You can use this example policy script:

    > **Example**

    ```
    function removeEndingZeros(b) {
      var len = b.length;
      while (len>0 && b[len-1] == 0) {
        len--;
      }
      var result = java.lang.reflect.Array.newInstance(java.lang.Byte.TYPE,
    len);
      java.lang.System.arraycopy(b, 0, result, 0, len);
      return result;
    }
    function hex2String(hex) {
    ```

```
  var b = new java.math.BigInteger(hex, 16).toByteArray();
  b = removeEndingZeros(b);
  var s = new java.lang.String(b, "ASCII");
  return s;
}
function searchUserByAttribute(name, value) {
  var searchFilter =
com.sap.security.api.UMFactory.getUserFactory().getUserSearchFilter();

searchFilter.setSearchAttribute(com.sap.security.api.IPrincipal.DEFAULT_
NAMESPACE, name, value,
com.sap.security.api.ISearchAttribute.EQUALS_OPERATOR, false);
  var searchResult =
com.sap.security.api.UMFactory.getUserFactory().searchUsers(searchFilter
);
  var uniqueID = (searchResult.size() == 1) ? searchResult.next() :
null;
  return uniqueID;
}
function onUserIdentified(config, context, result) {
  var OTP = "otp";
  var logger = context.getLogger();
  var authnMethod = context.getLoginInfo().getAuthenticationMethod();
  var principalName = context.getLoginInfo().getPrincipal().getName();
  var deviceUser =
com.sap.security.api.UMFactory.getUserFactory().getUserByLogonID(princip
alName);
  if (!context.getHttpClientContext().isUrlSignatureValid("signature"))
{
    result.abortLogin("Access denied");
    return;
  }
  var tagId = context.getHttpClientContext().getParameter("nfctagid");
  if (tagId == null) {
    result.abortLogin("Access denied");
    return;
  }
  var uniqueID = searchUserByAttribute("tagid", tagId);
  if (uniqueID == null) {
    result.abortLogin("Access denied");
    return;
  }
  var user =
com.sap.security.api.UMFactory.getUserFactory().getUser(uniqueID);
  var newPrincipal = context.createPrincipal(user.getUniqueName(), OTP);
  result.setPrincipal(newPrincipal);
}
function onFirstStageLogin(config, context, result) {
  result.doNotRequireSecondFactor();
  config.setProperty("tfa.require.user.confirmation", "yes");
}
```

## Users Accessing Applications

Provided that the applications are added in SAP Authenticator, users can access them from their mobile devices through mobile SSO by choosing the *Log on with SAP Authenticator* link. The URL for this link has the following format: `sapauthenticator://<host>:<port>/<path>?j_username=[username]&j_passcode=[passcode]`.

> **Example**
>
> Donna Moore, as an administrator at company A, would like to allow the company users to log on to SAP Enterprise Portal with mobile SSO through SAP Authenticator. To do this, she adds SAP Enterprise Portal as an application under the *Device* tab in the *One-Time Password Administration* UI. Because Donna has set the OTP-related logon application, the link is shown on the logon page. In addition, Donna assigns the `OTP_ONLINE_USER` role to her users so they can use the online account setup.
>
> As a user, Michael sets up an account in SAP Authenticator. The account contains the SAP Enterprise Portal application because Michael has used the online account setup. He then opens the application in a browser on his mobile device and chooses the *Log on with SAP Authenticator* link on the logon page. SAP Authenticator processes the authentication request, and Michael logs on without having to enter his username and passcode.

### Related Information

Configuring TOTPLoginModule and RBALoginModule [page 8]
SAP Authenticator Mobile Application [page 95]
Configuring an OTP-Related Logon Application [page 38]

## 1.2.9 Opening 3rd Party Mobile Applications with Credentials from SAP Authenticator

You can use the credentials provided from SAP Authenticator to log on to a 3rd party mobile application (short, 3rd party application).

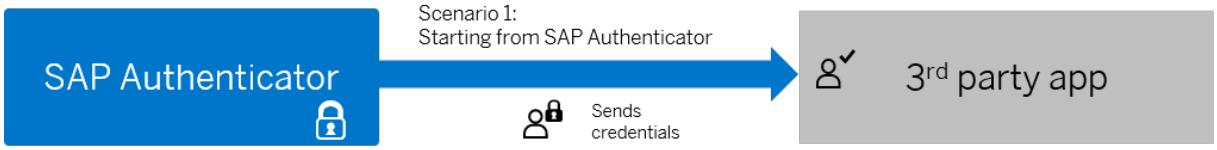
The 3rd party applications could be, for example, an application that runs on AS Java, and is protected with the `TOTPLoginModule`, or it can be an application running on a platform as a service provider, and the identity provider is an SAP identity provider protected with the `TOTPLoginModule`.

These are the two supported scenarios:

- Authenticate in a 3rd party application by starting from SAP Authenticator
- Authenticate in a 3rd party application by starting from the 3rd party application

### Authenticate in a 3rd Party Application by Starting from SAP Authenticator

You can use the credentials generated by SAP Authenticator for a particular user to open and log on to a 3rd party application.



### Prerequisites

- The 3rd party application handles its own custom scheme `<3rd party app scheme>`
- SAP Authenticator and the 3rd party application are installed on the same device

### Procedure

1. Set up the 3rd party application as a trusted application in SAP Authenticator. The URL of this application must have the following format:
   `<3rd party app scheme>://<host>:<port>/<path>?<3rd party username parameter>=[username]&<3rd party passcode parameter>=[passcode]`
   For more information on how to add an application, see Using Mobile Single Sign-On [page 44].
2. Open the 3rd party application by clicking on the application name in SAP Authenticator.

### How Does It Work

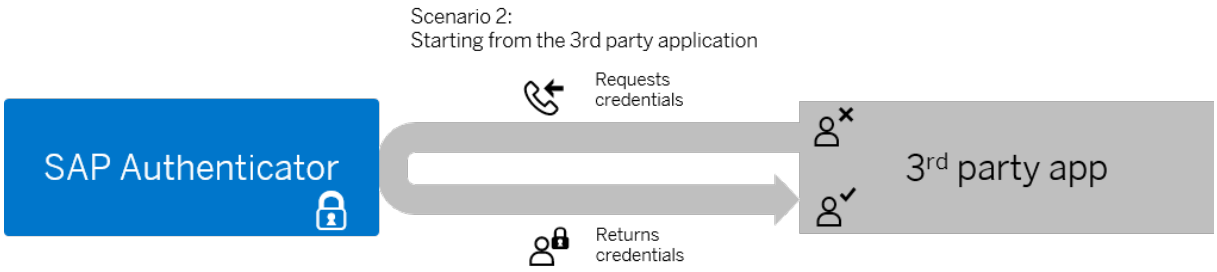
When you open the 3rd party application from SAP Authenticator, SAP Authenticator verifies whether one or more online accounts are activated. If there is only one account activated, SAP Authenticator replaces the placeholders for the credentials from the URL of the application (`[username]` and `[passcode]`) with the credentials for the corresponding account. If more accounts are activated in SAP Authenticator, the user will be asked to choose which account to be used.

The 3rd party application receives the URL with the credentials of the user, which are provided by SAP Authenticator to authenticate the user. The authentication of the user can happen against any application that can validate the username and passcode.

> **i** Note
>
> Keep in mind that the credentials provided by SAP Authenticator are time-based, and the user can authenticate only within the set validity period of the credentials.

## Authenticate in a 3rd Party Application by Starting from the 3rd Party Application

You can use SAP Authenticator to provide credentials for a user to log on to a 3rd party application, by starting from the 3rd party application. The communication between SAP Authenticator and the 3rd party application is done via custom URL schemes.



### Prerequisites

- The 3rd party application handles its own custom scheme `<3rd party app scheme>`
- SAP Authenticator and the 3rd party application are installed on the same device

### Procedure

1. Add the URL of the 3rd party application in SAP Authenticator. It must match the following format: `<3rd party app scheme>://<host>:<port>/`
   Once configured, the application is considered to be trusted application. For more information, see Using Mobile Single Sign-On [page 44].
   If the application is not set in SAP Authenticator, the user will get the following error:
   *To log on, add the site to SAP Authenticator or contact your administrator.*
2. On the 3rd party application, configure the 3rd party application URL, to have the following format: `sapauthenticator://<host>:<port>/<path>?<3rd party username parameter>=[username]&<3rd party passcode parameter>=[passcode]&x-callback-scheme=<3rd party app scheme>`
3. Save your configurations.

After completing these steps, SAP Authenticator and the 3rd party application are configured and you can use the credentials from SAP Authenticator to log on to the 3rd party application.

### How Does It Work

Depending on the individual implementation of the 3rd party application, it can be started in various ways. For the cases where credentials are required from SAP Authenticator, it is important that the URL of the 3rd party application has the following format:

```
sapauthenticator://<host>:<port>/<path>?<3rd party username parameter>=[username]&<3rd party passcode parameter>=[passcode]&x-callback-scheme=<3rd party app scheme>
```

SAP Authenticator recognizes the `sapauthenticator` scheme, takes the URL and verifies whether one or more online accounts are activated. If there is only one account activated, SAP Authenticator replaces the placeholders for the credentials from the URL of the application (`[username]` and `[passcode]`) with the credentials for the corresponding account. If more accounts are activated in SAP Authenticator, the user will be asked to choose which account to be used.

Once the URL contains the credentials, SAP Authenticator returns the following URL to the 3rd party application, by setting the `<3rd party app scheme>` parameter as scheme to call the 3rd party application:

```
<3rd party app scheme>://<host>:<port>/<path>?<3rd party username>=[replaced by username]&<3rd party passcode>=[replaced by passcode]
```

As a result, the 3rd party application receives the URL with the credentials of the user, which are provided by SAP Authenticator to authenticate the user. The authentication of the user can happen against any application that can validate the username and passcode.

## 1.2.10 Additional Settings

You can set other passcode properties in the One-Time Password (OTP) Administration user interface (UI), which are applicable for all OTP users and applications.

### Context

You can set the following additional properties:

- System name
  The system name property indicates the provider of the secret key.

  > **i Note**
  >
  > The default value is an empty string.

- Default validity in days
  This is the initial validity in days of the time-based one-time password (TOTP) secret key. It can be changed later under the *User Management* tab.

  > **i Note**
  >
  > The default value is 365

  > **i Note**
  >
  > You cannot set a new default validity for an already setup account because the default validity is taken into consideration only when a user account is initially set up.

- Expiration warning period
  This is the period in days before the secret key expiration. When this period starts, users receive warning messages prompting them to update their accounts.

  > **i Note**
  >
  > The default value is 14.

  Depending on the validity and expiration period set in the *One-Time Password Administration* UI, following operations are allowed:

| Secret Key Validity | Can use the passcode for authentication? | Can register new device on the *Mobile Device Setup* page? | Can disable device on the *Mobile Device Setup* page? |
|---|---|---|---|
| *Expiration Warning Period* is not reached | Yes | Yes | Yes |

| Secret Key Validity | Can use the passcode for authentication? | Can register new device on the *Mobile Device Setup* page? | Can disable device on the *Mobile Device Setup* page? |
|---|---|---|---|
| During *Expiration Warning Period* | Yes<br>Warning is shown | No<br>You can disable the registered device and then create a new registration. | Yes |
| Secret key has expired | No<br>Error message is shown that the registration has expired | No<br>You can disable the registered device and then create a new registration. | Yes<br>For security reasons, passcode of expired device is required |

- Passcode length
  You have two alternatives for the time-based one-time password (TOTP) length for the SAP Authenticator mobile application: 6 digits and 8 digits.

  > **i Note**
  >
  > The default value is 8 digits.

- Digest algorithm
  You can define which digest algorithm to be used for the generation of passcodes.

  > **i Note**
  >
  > The default value is SHA-512.

- Maximum number of failed logon attempts
  The number of consecutive failed logon attempts after which the user account is locked.

  > **i Note**
  >
  > The default value is 5.

- Automatic unlock time
  The period in minutes that a user cannot log on with a passcode because he or she has exceeded the allowed failed logon attempts.

  > **i Note**
  >
  > The default value is 60.

- Show secret key
  The setup page shows the secret key or the setup URL (for online account setup) in plain text when the user selects the checkbox below the QR code on the *Mobile Device Setup* UI.

  > **i Note**
  >
  > This property is not selected by default.

- Installation section shown on the *Mobile Device Setup* UI

Provided the option is enabled, the UI displays the default installation section if the URL path of the custom installation section is an empty string. You can also set a custom installation section by entering custom URL path and height.

> **i** Note
>
> By default this option is enabled, the URL path is set to an empty string, and the section's height is 300 pixels.

For the configuration of the additional properties, proceed as follows:

### Procedure

1. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
2. Choose the *Settings* tab.
3. Choose the *Edit* button.
4. Configure the additional properties.
5. Save your settings.

## 1.2.11 Repairing Inconsistencies from SAP NetWeaver Administrator

You can repair some misconfiguration in the SAP NetWeaver Administrator related to one-time password (OTP) authentication.

### Context

SAP NetWeaver Administrator (NWA) is not to be used for the configuration of the OTP properties, but the *One-Time Password Administration* UI. If you configure such properties in SAP NWA, you will not be able to see any new settings of these properties in the *One-Time Password Administration* UI until you fix the inconsistencies. When there are such inconsistencies, you see a separate section at the top of the UI with a *Repair* button.

### Procedure

1. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
2. Choose the *Settings* tab.
3. Choose the *Repair* button.

   Any inconsistent settings in the *One-Time Password Administration* UI show up after the repair.

## 1.2.12 SAP Single Sign-On Configuration for Network Edge Authentication

Network Edge Authentication (NEA) is based on SAP Web Dispatcher and SAP Single Sign-On products. It provides integrated, simple and secure Web access control for SAP solutions.

### Prerequisites

- SSOAUTHLIB.sca for SAP Single Sign-On 3.0 SP01 or higher is installed. For more information, see One-Time Password Authentication Installation and Upgrade Guide [page 4].
- SAP Web Dispatcher version 7.51 SP00 or higher is installed and configured. Make sure that you have set the relative path */nea/v1/authenticate* as part of the absolute URL for the *AUTH_SERVICE* subparameter. For more information, see *Network Edge Authentication* and *SAP Web Dispatcher Configuration for Network Edge Authentication* in the SAP Web Dispatcher documentation at http://help.sap.com/nw751abap, *Application Help* > *SAP NetWeaver Library: Function-Oriented View* > *Application Server Infrastructure - ABAP* > *Components of SAP NetWeaver Application Server for ABAP* .
- If you want to use certificates for SSO tokens, for example if you configure X.509 certificate as SSO token type in the SAP Web Dispatcher, you have to install and configure the Secure Login Server, version 3.0 SP01 or higher.

### How Does It Work

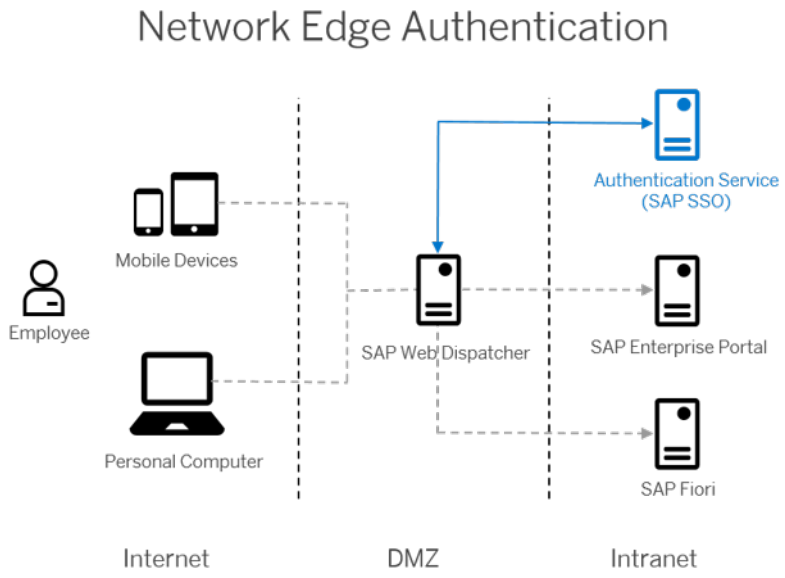
When a user tries to access a back-end system, the SAP Web Dispatcher forwards the request to the authentication service provided by SAP Single Sign-On. The authentication service triggers the authentication process and after successful authentication, the SSO service issues an SSO token to be used by the SAP Web Dispatcher for accessing the back-end systems. These tokens could be X.509 certificates or logon tickets.

If logon tickets are used as SSO token type, no additional configuration is required for the Network Edge Authentication service. You only need to do the standard trust configuration between the system where the authentication service is installed (as issuer of the logon ticket), and the back-end system (as receiver of the logon ticket).

If X.509 certificates are used as SSO token type, you have to configure an authentication profile in the *Secure Login Administration Console*, as well as the name of this profile as a value for the property **`x509_sls_profile_name`** for the **`sap.com/sso~nea~ear*nea_v1_authenticate`** authentication stack on SAP NetWeaver Administrator.

> ⚠ Caution
>
> When configuring the profile for issuing X.509 certificates in the *Secure Login Administration Console*, you can only use the **(AUTH:USERID)** variable. The other variables are not supported by the Network Edge Authentication service. If you set any of them, the certificate issuing process will fail.

For more information, see the SAP Web Dispatcher documentation and the Secure Login Implementation Guide.




The authentication stack used for the NEA authentication service is **sap.com/sso~nea~ear*nea_v1_authenticate**. By default, the NEA authentication service uses the *ticket* authentication stack template. You can change this setting by reconfiguring the authentication stack of the NEA authentication service. To do so, proceed as follows:

1. Open SAP NetWeaver Administrator at *http(s)://<host>:<port>/nwa*.
2. Choose *Configuration* > *Authentication and Single Sign-On* > *Authentication* > *Components* > .
3. Search for **sap.com/sso~nea~ear*nea_v1_authenticate** of type **Web**.
4. Edit the login modules on the *Authentication Stack* tab.
   You can set a two-factor or risk-based authentication using the *TOTPLoginModule*, for example.

For more information on how to configure the *TOTPLoginModule*, see Configuring TOTPLoginModule and RBALoginModule [page 8].

## 1.3 One-Time Password Authentication Developer Guide

In addition to the administration settings, you can develop various scripts to control the authentication process. The scripts are written in JavaScript language.

All policy scripts are developed in the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts`. They can be then executed from the *One-Time Password Administration* UI. Only the activated versions of enabled policy scripts of type *Procedure* will be visible in the *One-Time Password Administration* UI. For more information, see Policy Scripts Implementation Guide.

> **i Note**
>
> To create policy scripts in the *Policy Script Administration Console*, you need the *RBA_POLICY_WRITE* user management engine (UME) action or RBA_POLICY_ADMIN UME role assigned to your user. For more information, see Policy Scripts Installation Guide.

There are two ways of implementing the policy scripts for OTP authentication: By creating policy scripts, and by reusing a policy script of type *Library* in another policy script.

> **i** Note
>
> You can find example scripts in SAP Note 2225027 and in the tutorials of this guide.

## Creating Policy Scripts

For the following scenarios you can develop a policy script in the *Policy Script Administration Console* and execute it from the *One-Time Password Administration* UI:

- Controlling the authentication process for risk-based authentication
  You can develop a script that defines the authentication method at runtime in accordance with various conditions. You can make a second factor necessary for users logging on from certain locations for example, but only one factor for other locations.
- Setting user permissions for context-based authorization
  By using a simple script you can configure the permissions for users to some functionality, without changing the user authorization assignments. For example, you can limit this access when the users authenticate from outside the corporate network. This functionality will be available for the users when they are working from the corporate network.
- Configuring logs and traces
  You can call methods for OTP logs and traces. For more information, see Policy Script Functions and Methods [page 57].
- Calling OTP methods in your script functions
  In your script, you can call methods in the *onInitialize(...)*, *onFirstStageLogin(...)*, *onSecondStageLogin(...)*, and other functions. For more information, see Policy Script Functions and Methods [page 57].
- Configuring two-factor authentication to use passcodes sent using out-of-band methods.
  You can develop scripts for the passcode to be sent via SMS, e-mail or other non-standard methods. For more information, see Develop a Script for Risk-Based Authentication [page 83].
- Configuring validation of a passcode generated by a third-party passcode provider.
  You can develop a policy script that checks whether a passcode is valid or not. For more information, see Develop a Script for External Passcode Validation [page 90].
- Using the policy script for an application
  You can control the authentication for a specific policy configuration (application) by developing a policy script. In this case, you have to specify the policy script for the *Policy* option of `TOTPLoginModule`.

## Reusing Policy Scripts

- You can call a *Library* policy script in another policy script of type *Procedure* with the *include* directive. For more information, see Reusing Policy Scripts.

> **Example**
>
> After creating a *mail* policy script in the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts`, you can use the following:
>
> ```
> #include "mail";
> ...
> ```

More Information

- Configuring TOTPLoginModule and RBALoginModule [page 8]
- Working with Policy Scripts

## 1.3.1 Policy Script Functions and Methods

You can use pre-defined methods in your JavaScript when developing a policy script.

### Policy Script Functions

We recommend using the following functions for the policy script methods:

```
/*
 * @param config - object of type PolicyConfiguration
 * @param context - object of type PolicyContext;
 */
function onInitialize(config, context) {
}
/*
* @param config - object of type PolicyConfiguration
* @param context - object of type PolicyContext;
* @param result - object of type UserIdentifiedLoginResult;
*/
function onUserIdentified(config, context, result) {
}
/*
 * @param config - object of type PolicyConfiguration
 * @param context - object of type PolicyContext;
 * @param result - object of type FirstStageLoginResult;
 */
function onFirstStageLogin(config, context, result) {
}
/*
 * @param config - object of type PolicyConfiguration
 * @param context - object of type PolicyContext;
 * @param result - object of type SecondStageLoginResult;
 */
function onSecondStageLogin(config, context, result) {
}
/*
 * @param config - object of type PolicyConfiguration
 * @param context - object of type PolicyContext;
 */
function onAbort(config, context) {
}
/*
```

```
 * @param config - object of type PolicyConfiguration
 * @param context - object of type PolicyContext;
 */
function onLogout(config, context) {
}
```

You can use the following functions for a context-based authorization:

```
/*
 * @param config - object of type PolicyConfiguration
 * @param context - object of type PolicyContext;
 * @param result - object of type FirstStageLoginResult;
 */
function onLimitedAccessSelected(config, context, result) {
}
/*
 * @param config - object of type PolicyConfiguration
 * @param context - object of type PolicyContext;
 * @param result - object of type FirstStageLoginResult;
 */
function onFullAccessSelected(config, context, result) {
}
```

You can also use the following function for external passcode validation:

```
/*
 * @param config - object of type PolicyConfiguration
 * @param context - object of type PolicyContext;
 * @param result - object of type PasscodeValidationLoginResult;
 * @param username - object of type String;
 * @param passcode - object of type String;
 */
function validatePasscode(config, context, result, username, passcode) {
}
```

## Policy Script Methods

| Object | Method | Description |
| --- | --- | --- |
| *PolicyConfiguration* | `getProperty(name)` | Returns the value of the policy configuration property.<br>The `name` parameter is of type `String` and specifies the name of the requested property. |

| Object | Method | Description |
| --- | --- | --- |
| | `setProperty(name, value)` | Sets a property of the policy configuration.<br><br>The `name` and `value` parameters are of type `String` and specify the name of the property and the assigned value.<br><br>**Example**<br>`if (context.getHttpClientContext().getClientIP() == "10.11.12.13") {`<br>`config.setProperty("tfa.first.factor.login.module", "SPNegoLoginModule");`<br>`} else {`<br>`config.setProperty("tfa.first.factor.login.module", "BasicPasswordLoginModule");`<br>`}` |
| *PolicyContext* | `getLogger()` | Returns a `LogAccessor` object for the policy context. The returned object is used for logging and tracing.<br><br>**Example**<br>`var logger = context.getLogger();`<br>`var now = new Date();`<br>`logger.traceDebug("Logon time: " + now);` |
| | `getNaming()` | Returns an `javax.naming.InitialContext` object used for looking up with Java Naming and Directory Interface (JNDI). |

| Object | Method | Description |
| --- | --- | --- |
| | `getHttpClientContext()` | Returns an `HttpClientContext` object.<br>**Example**<br>`if (context.getHttpClientContext().getClientIP() == "10.11.12.13") { config.setProperty("tfa.first.factor.login.module", "SPNegoLoginModule"); } else { config.setProperty("tfa.first.factor.login.module", "BasicPasswordLoginModule"); }` |
| | `getRsaContext()` | Returns an `RSAContext` object.<br>**Example**<br>`var rsaContext = context.getRsaContext(); if (rsaContext.validatePasscode(username, passcode) == rsaContext.VALID_PASSCODE){ result.setValidationSuccessful(); }` |

| Object | Method | Description |
| --- | --- | --- |
| | `getRadiusContext()` | Returns an `RADIUSContext` object.<br><br>**Example**<br>`var radiusContext = context.getRadiusContext();`<br>`if (radiusContext.validatePasscode(username, passcode) == radiusContext.VALID_PASSCODE){`<br><br>`result.setValidationSuccessful();`<br>`}` |
| | `getLoginInfo()` | Returns a `LoginInfo` object used for logon information such as logon user, authentication method, and validity of user account.<br><br>**Note**<br>The method returns *null* when the first stage logon was not completed.<br><br>**Example**<br>`var loginInfo = context.getLoginInfo();`<br>`var user = loginInfo.getUser();` |

| Object | Method | Description |
| --- | --- | --- |
| | `createVirtualUserPrincipalBuilder(name, authenticationMethod)` | Returns `Builder` for *VirtualUserPrincipal*. The principal has name and authentication method as specified as parameters.<br><br>By calling the `build()` method of this builder, a *VirtualUserPrincipal* is returned and can be set in the result of the `onUserIndetified` method. The methods of this builder are:<br><br>• `withGroups(<group1_name>, <group2_name>, …)` – one or more parameters of type `String`<br>• `withRoles(<role1_name>, <role2_name>, …)` – one or more parameters of type `String`<br>• `withAttribute(namespace, name, value)` – parameters are of type `String`<br>• `withAttributeDefaultNamespace(name, value)` – the default namespace will be used ("com.sap.security.core.usermanagement"); parameters are of type `String`<br>• `withBinaryAttribute(namespace, name, value)` – namespace and name are of type `String`, value is of type `byte`<br>• `withBinaryAttributeDefaultNamespace(name, value)` – the default namespace will be used ("com.sap.security.core.usermanagement"); namespace and name are of type `String`, value is of type `byte`<br><br>**Example**<br><br>`function onUserIndetified(config, context, result) {` |

| Object | Method | Description |
| --- | --- | --- |
| | | ```<br>var virtualUserPrincipalBuilder = context.createVirtualUserPrincipalBuilder("username" , "virtual_user");<br>var virtualUserPrincipal = virtualUserPrincipalBuilder.withGroups("group1", "group2").withRoles("role1", "role2").withAttribute("com.sap.security.core.usermanagement", "email", john.doe@example.com).withAttributeDefaultNamespace("phone", "+359000000000").build();<br>result.setPrincipal(virtualUserPrincipal);<br>}<br>``` |
| | `createPrincipal(name, authenticationMethod)` | Returns `Principal` with name and authentication method as specified as parameters.<br>**Example**<br>```<br>context.createPrincipal("username", "client_cert");<br>``` |

| Object | Method | Description |
| --- | --- | --- |
| *FirstStageLoginResult* | `setRandomPasscode(length, validity, maximumFailedAttempts, message)` | Generates a random passcode and returns the passcode as `String`.<br><br>Parameters:<br>• `length` - passcode length, of type `int`<br>• `validity` - passcode validity in minutes, of type `int`<br>• `maximumFailedAttempts` - the allowed passcode logon attempts before the account is locked, of type `int`<br>• `message` - the text shown on the logon page, of type `String`.<br><br>**Example**<br>`var passcode = result.setRandomPasscode(10, 15, 5, "Passcode sent via SMS; enter the passcode to log on");` |
| | `setPasscode(passcode, validity, maximumFailedAttempts, message)` | Sets a generated passcode.<br><br>Parameters:<br>• `passcode` - the generated passcode, of type `String`<br>• `validity` - passcode validity in days, of type `int`<br>• `maximumFailedAttempts` - the allowed passcode logon attempts before the account is locked, of type `int`<br>• `message` - the message text shown on the logon page, of type `String`. |

| Object | Method | Description |
| --- | --- | --- |
| | `doNotRequireSecondFactor()` | Sets the application not to require a second factor after first stage logon.<br><br>**Example**<br>The script may not require a second factor if the user is not a manager.<br><br>`if (! user.isMemberOfGroup( 'GRUP.PRIVATE_DATASOURCE.un:Managers', true)) { result.doNotRequireSecondFactor(); }` |
| | `setOptionalSecondFactor()` | Allows the user to decide whether to use a second factor.<br><br>If you use this method, the application's logon page displays the default text:<br><br>`Select your access permission to log on:`<br><br>`(*) Limited Access Permissions (Log on directly)`<br><br>`( ) Full Access Permissions (Log on with a passcode)`<br><br>`[Continue] [Cancel]` |

| Object | Method | Description |
| --- | --- | --- |
| | `setOptionalSecondFactor(message, firstOption, secondOption, defaultSelection)` | Allows the user to decide whether to use a second factor, and customizes the default text according to the specified parameters.<br><br>Parameters:<br><br>• `message`, `firstOption`, and `secondOption` parameters are of type `String`.<br>• `defaultSelection` is a parameter with two values: *result.DEFAULT_SELECTION_LIMITED_ACCESS*, and *result.DEFAULT_SELECTION_FULL_ACCESS*.<br><br>**Example**<br><br>The following example selects the first option by default.<br><br>`result.setOptionalSecondFactor("Select your access permissions to log on:", "Limited Permissions", "Manager's Permissions", result.DEFAULT_SELECTION_LIMITED_ACCESS);` |

| Object | Method | Description |
| --- | --- | --- |
| | `setAuthorizationScopes(scopes,roleWithoutScopes)` | Users are assigned roles in the user management engine (UME). Some of the roles can also contain scopes. The `setAuthorizationScopes(scopes, roleWithoutScopes)` method dynamically assigns to users those roles which comply with the specified parameters.<br><br>Parameters:<br><br>• `scopes` - a string array of scopes. Users receive their UME roles that contain the specified scopes. If no scopes are specified in the brackets, the `roleWithoutScopes` parameter is taken into consideration.<br>• `roleWithoutScopes` - a parameter with two values:<br>  • *result.GRANT_ROLES_WITHOUT_SCOPES* - the UME roles without scopes are granted.<br>  • *result.DENY_ROLES_WITHOUT_SCOPES* - the UME roles without scopes are not granted.<br><br>**Example**<br><br>The following example grants full access to users with UME roles containing the "FULL_ACCESS" scope and with UME roles that do not have scopes.<br><br>`result.setAuthorizationScopes(["FULL_ACCESS"], result.GRANT_ROLES_WITHOUT_SCOPES);` |

| Object | Method | Description |
| --- | --- | --- |
| | `abortLogin(message)` | Resets the authentication process, that is, any previous results stored in the session are removed.<br><br>The `message` parameter is used to specify the message that the user receives when the logon is terminated.<br><br>**Example**<br>`result.abortLogin("You are not allowed to log on outside of the working hours");` |
| *SecondStageLoginResult* | `abortSecondStage(message)` | Aborts only the logon for the second stage, but not the whole authentication process.<br><br>The `message` parameter is used to specify the message that the user receives when the logon is terminated. |

| Object | Method | Description |
| --- | --- | --- |
| | `setAuthorizationScopes(scopes,roleWithoutScopes)` | Users are assigned roles in the user management engine (UME). Some of the roles can also contain scopes. The `setAuthorizationScopes(scopes, roleWithoutScopes)` method dynamically assigns to users those roles which comply with the specified parameters.<br><br>Parameters:<br><br>• `scopes` - a string array of scopes. Users receive their UME roles that contain the specified scopes. If no scopes are specified in the brackets, the `roleWithoutScopes` parameter is taken into consideration.<br>• `roleWithoutScopes` - a parameter with two values:<br>  • *result.GRANT_ROLES_WITHOUT_SCOPES* - users are granted all their UME roles without scopes.<br>  • *result.DENY_ROLES_WITHOUT_SCOPES* - users are not granted all their UME roles without scopes.<br><br>**Example**<br><br>The following example grants full access to users with UME roles containing the "FULL_ACCESS" scope and with UME roles that do not have scopes.<br><br>`result.setAuthorizationScopes(["FULL_ACCESS"], result.GRANT_ROLES_WITHOUT_SCOPES);` |

| Object | Method | Description |
| --- | --- | --- |
| | `abortLogin(message)` | Resets the authentication process, that is, any previous results stored in the session are removed.<br>The `message` parameter is used to specify the message that the user receives when the logon is terminated. |
| *UserIdentifiedLoginResult* | `setPrincipal(principal)` | Sets `Principal` as specified by the `principal` parameter.<br>The `message` parameter is used to specify the message that the user receives when the logon is terminated. |

| Object | Method | Description |
| --- | --- | --- |
| | `setAuthorizationScopes(scopes,roleWithoutScopes)` | Users are assigned roles in the user management engine (UME). Some of the roles can also contain scopes. The `setAuthorizationScopes(scopes, roleWithoutScopes)` method dynamically assigns to users those roles which comply with the specified parameters.<br><br>Parameters:<br><br>• `scopes` - a string array of scopes. Users receive their UME roles that contain the specified scopes. If no scopes are specified in the brackets, the `roleWithoutScopes` parameter is taken into consideration.<br>• `roleWithoutScopes` - a parameter with two values:<br>  • `result.GRANT_ROLES_WITHOUT_SCOPES` - users are granted all their UME roles without scopes<br>  • `result.DENY_ROLES_WITHOUT_SCOPES` - users are not granted all their UME roles without scopes<br><br>**Example**<br><br>The following example grants full access to users with UME roles containing the "FULL_ACCESS" scope and with UME roles that do not have scopes.<br><br>`result.setAuthorizationScopes(["FULL_ACCESS"], result.GRANT_ROLES_WITHOUT_SCOPES);` |

| Object | Method | Description |
| --- | --- | --- |
| | `abortLogin(message)` | Resets the authentication process, that is, any previous results stored in the session are removed.<br>The `message` parameter is used to specify the message that the user receives when the logon is terminated. |
| *Logger* | `traceError(message)` | The trace showing the `message` text in the log viewer of SAP NetWeaver Administrator with severity level *Error*.<br>The `message` parameter is of type `String`. |
| | `traceError(message, throwable)` | The trace showing the `message` text in the log viewer of SAP NetWeaver Administrator with severity level *Error*.<br>Parameters:<br>• message - of type `String`<br>• throwable - indicates the thrown exception, of type `Object` |
| | `traceWarning(message)` | The trace showing the `message` text in the log viewer of SAP NetWeaver Administrator with severity level *Warning*.<br>The `message` parameter is of type `String`. |
| | `traceWarning(message, throwable)` | The trace showing the `message` text in the log viewer of SAP NetWeaver Administrator with severity level *Warning*.<br>Parameters:<br>• message - of type `String`<br>• throwable - indicates the thrown exception, of type `Object` |
| | `traceInfo(message)` | The trace showing the `message` text in the log viewer of SAP NetWeaver Administrator with severity level *Info*.<br>The `message` parameter is of type `String`. |

| Object | Method | Description |
| --- | --- | --- |
| | `traceDebug(message)` | The trace showing the `message` text in the log viewer of SAP NetWeaver Administrator with severity level *Debug*.<br>The `message` parameter is of type `String`. |
| | `traceDebug(message, throwable)` | The trace showing the `message` text in the log viewer of SAP NetWeaver Administrator with severity level *Debug*.<br>Parameters:<br>• message - of type `String`<br>• throwable - indicates the thrown exception, of type `Object` |
| | `logError(message)` | The log showing the `message` text in the log viewer of SAP NetWeaver Administrator with severity level *Error*.<br>The `message` parameter is of type `String`. |
| | `logWarning(message)` | The log showing the `message` text in the log viewer of SAP NetWeaver Administrator with severity level *Warning*.<br>The `message` parameter is of type `String`. |
| | `logInfo(message)` | The log showing the `message` text in the log viewer of SAP NetWeaver Administrator with severity level *Info*.<br>The `message` parameter is of type `String`. |
| *HttpClientContext* | `getClientIP()` | Returns the IP of the client that the request is coming from. The returned value is `String`. |

| Object | Method | Description |
| --- | --- | --- |
| | `getHeader(name)` | Returns the header value as `String`.<br><br>The `name` parameter is of type `String`.<br><br>**Example**<br>To return the value of the `Host` header, call the following method:<br>`getHeader("Host")` |
| | `setHeader(name, value)` | Creates or sets a header with the specified value.<br><br>The `name` and `value` parameters are of type `String`. |
| | `removeHeader(name)` | Removes the specified header.<br><br>The `name` parameter is of type `String`. |
| | `getCookie(name)` | Returns the cookie value as `String`. |
| | `setPersistentCookie(name, value, path, domain, validity, httpOnly, secure)` | Creates or sets a persistent cookie with the specified attributes.<br><br>Parameters:<br>• The `name`, `value`, `path`, and `domain` parameters are of type `String`.<br>• The `validity` parameter indicates the cookie's validity in seconds and is of type `long`.<br>• The `httpOnly` and `secure` parameters are of type `boolean`. |
| | `setSessionCookie(name, value, path, domain, httpOnly, secure)` | Creates or sets a session cookie with the specified attributes.<br><br>Parameters:<br>• The `name`, `value`, `path`, and `domain` parameters are of type `String`.<br>• The `httpOnly` and `secure` parameters are of type `Boolean`. |

| Object | Method | Description |
| --- | --- | --- |
| | `removeCookie(name, path, domain)` | Removes a session or persistent cookie with the specified attributes.<br>The `name`, `path`, and `domain` parameters are of type `String`. |
| | `isSecure()` | Checks if the cookie is secure and returns `boolean`: true if the cookie is secure, and false otherwise. |
| | `getParameter(name)` | Returns the first value of the specified parameter in the Request as `String`.<br>The `name` parameter is of type `String`. |
| | `getParameterValues(name)` | Returns the values of the specified parameter in the Request as a `String[]` array.<br>The `name` parameter is of type `String`. |
| | `getRequestAttribute(name)` | Returns the value of the specified attribute in the Request as `String`.<br>The `name` parameter is of type `String`. |
| | `setRequestAttribute(name, value)` | Creates or sets the attribute in the Request with the specified value.<br>The `name` and `value` parameters are of type `String`. |
| | `getSessionAttribute(name)` | Returns the value of the session attribute as `String`.<br>The `name` parameter is of type `String`. |
| | `setSessionAttribute(name, value)` | Creates or sets the session attribute with the specified value.<br>The `name` and `value` parameters are of type `String`. |

| Object | Method | Description |
| --- | --- | --- |
| | `removeSessionAttribute(name)` | Removes the specified session attribute.<br>The `name` parameter is of type `String`. |
| | `getLocale()` | Returns a `Locale` object, which shows where the request comes from. |
| | `getClientCertificateChain()` | Returns a chain of the trusted client certificates as a `Certificate[]` array. |
| *RSAContext* | `validatePasscode(passcode)` | Validates the passcode and returns a `String` value showing if the user logon is successful. The returned value corresponds to one of the following:<br>• `rsaContext.VALID_PASSCODE`<br>The pascode is valid and the logon is successful.<br>• `rsaContext.INVALID_PASSCODE`<br>The passcode is not correct and the authentication failed.<br>• `rsaContext.VALID_NEXT_PASSCODE_REQUIRED`<br>The entered passcode is valid, but a second one is required for logon.<br>• `rsaContext.RSA_SERVICE_UNAVAILABLE`<br>The used RSA service is unavailable for passcode validation.<br>The `passcode` parameter is of type `String`. |

| Object | Method | Description |
| --- | --- | --- |
| | `validatePasscode(username, passcode)` | Validates the username and passcode and returns a `String` value showing if the user logon is successful. The returned value corresponds to one of the following: `rsaContext.VALID_PASSCODE`, `rsaContext.INVALID_PASSCODE`, `rsaContext.VALID_NEXT_PASSCODE_REQUIRED` or `rsaContext.RSA_SERVICE_UNAVAILABLE`.<br><br>The `username` and `passcode` parameters are of type `String`. |
| | `setRADIUSDestination(destinationName)` | Sets the name of an existing SLS destination of type `RADIUS Destination`. For more information, see Managing Destinations.<br><br>The `destinationName` parameter is of type `String`. |

| Object | Method | Description |
| --- | --- | --- |
| *RADIUSContext* | `validatePasscode(passcode)` | Validates the passcode and returns a `String` value showing if the user logon is successful. The returned value corresponds to one of the following:<br>• `radiusContext.VALID_PASSCODE`<br>The pascode is valid and the logon is successful.<br>• `radiusContext.INVALID_PASSCODE`<br>The passcode is not correct and the authentication failed.<br>• `radiusContext.VALID_NEXT_PASSCODE_REQUIRED`<br>The entered passcode is valid, but a second one is required for logon.<br>• `radiusContext.RSA_SERVICE_UNAVAILABLE`<br>The used RSA service is unavailable for passcode validation.<br>The `passcode` parameter is of type `String`. |
| | `validatePasscode(username, passcode)` | Validates the username and passcode and returns a `String` value showing if the user logon is successful. The returned value corresponds to one of the following: `radiusContext.VALID_PASSCODE`, `radiusContext.INVALID_PASSCODE`, `radiusContext.VALID_NEXT_PASSCODE_REQUIRED` or `radiusContext.RSA_SERVICE_UNAVAILABLE`.<br>The `username` and `passcode` parameters are of type `String`. |

| Object | Method | Description |
| --- | --- | --- |
| | `setRADIUSDestination(destinationName)` | Sets the name of an existing SLS destination of type `RADIUS Destination`. For more information, see Managing Destinations.<br><br>The `destinationName` parameter is of type `String`. |
| *PasscodeValidationLoginResult* | `setValidationSuccessful()` | The following methods define how the result is shown in the UI.<br><br>Sets the result of the validation as successful. |
| | `setValidationFailure()` | Sets the result of the validation as not successful and no messages are shown to the user. |
| | `setValidationFailureWithErrorMessage(message)` | Sets the result of the validation as not successful and an error message is shown to the user. The `message` parameter used for the error message is of type `String`. |
| | `setValidationFailureWithWarningMessage(message)` | Sets the result of the validation as not successful and a warning message is shown to the user. The `message` parameter used for the warning message is of type `String`. |
| | `setValidationFailureWithInfoMessage(message)` | Sets the result of the validation as not successful and an info message is shown to the user. The `message` parameter used for the info message is of type `String`. |

| Object | Method | Description |
| --- | --- | --- |
| *LoginInfo* | `getUser()` | Returns an `IUser` object containing information about the user. For more information about this object, see Uses of Interface com.sap.security.api.IUser<br>**Example**<br>`var loginInfo = context.getLoginInfo(); var user = loginInfo.getUser(); if (! user.isMemberOfGroup('GRUP.PRIVATE_DATASOURCE.un:Managers', true)) { result.doNotRequireSecondFactor();` |
| | `getTOTPInfo()` | Returns a `TOTPInfo` object.<br>**Example**<br>`var loginInfo = context.getLoginInfo(); var totpInfo = loginInfo.getTOTPInfo(); if (totpInfo.getStatus() == totpInfo.DISABLED){ ... }` |
| | `getAuthenticationMethod()` | Returns the used authentication method as `String`.<br>Possible returned values: *password*, *client-cert*, *spnego*, *otp*, *password + otp*, *otp + password*, *otp + cookie*, *spnego + otp*, *unknown*, *unknown + otp*, or others. |

| Object | Method | Description |
| --- | --- | --- |
| | `getPrincipal()` | Returns a `Principal` object.<br>**Example**<br>`var loginInfo = context.getLoginInfo();`<br>`var principal = loginInfo.getPrincipal();`<br>`if (principal instanceof com.sap.engine.lib.security.ClientCertificatePrincipal) {`<br>`result.doNotRequireSecondFactor();`<br>`}` |
| *TOTPInfo* | `getStatus()` | Returns a string signifying the status of the user account.<br>The status is equivalent to a value of one of the following constants: *TOTPInfo.ENABLED*, *TOTPInfo.DISABLED*, *TOTPInfo.EXPIRED*, *TOTPInfo.SOON_TO_EXPIRE*, *TOTPInfo.LOCKED*. |
| | `getPasscodeLength()` | Returns the length of the current passcode as `int`. |
| | `getDigestAlgorithm()` | Returns the digest algorithm as `String` used to generate the passcode.<br>The digest algorithm is equivalent to a value of one the following constants: `TOTPInfo.SHA1`, `TOTPInfo.SHA256`, `TOTPInfo.SHA512`. |
| | `getSecretKeyValidity()` | Returns the validity in minutes of the time-based one-time password (TOTP) secret key as `int`, which specifies the user account validity. |

| Object | Method | Description |
| --- | --- | --- |
| | `getLockedPeriod()` | Returns the period in minutes during which the passcode is locked. The returned value is `int`. |

### Related Information

Log Viewer

## 1.3.2 Configuring the One-Time Password Administration UI for Policy Scripts

In the *One-Time Password Administration* UI, you can call a policy script that controls the authentication process for all applications that use `TOTPLoginModule` with *otp&pwd* mode, provided the login module option for policy scripts is not assigned. This script must be created and activated in the *Policy Script Administration Console*. For more information about creating policy scripts, see the Policy Scripts Implementation Guide.

> **i** Note
>
> If you want to create a script for a specific policy configuration (application), you can overwrite the policy script set in the *One-Time Password Administration* UI with the `policy` option for the `TOTPLoginModule`. For more information about the login module configuration, see Related Information.

### Procedure

1. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
2. Choose the *Settings* tab.

   > **i** Note
   >
   > To allow applications to use your policy script, make sure that the *Policy* checkbox under the *Two-Factor Authentication* section is selected.
3. Choose the *Policy Script...* button under the *Two-Factor Authentication* section.
4. Choose one of the following options:
   - Select a policy script from the dropdown list. Only the activated versions of enabled policy scripts of type *Procedure* are visible in the dropdown list.
   - Choose *Manage Policy Scripts* to go to the *Policy Script Administration Console* and create a new script. Then reload the policy scripts and select the newly created policy script from the dropdown list in the *Policy* window in the *One-Time Password Administration* UI.

5. Save your configuration.

### Related Information

Policy Scripts Implementation Guide
Configuring TOTPLoginModule and RBALoginModule [page 8]
Policy Script Functions and Methods [page 57]

## 1.3.3 Tutorials

### 1.3.3.1 Develop a Script for Risk-Based Authentication

This tutorial helps you develop a script for controlling risk-based authentication and out-of-band methods such as SMS and email.

### Prerequisites

- You have configured your policy configuration (application) to use the `TOTPLoginModule` with *otp&pwd* mode. For more information, see Configuring TOTPLoginModule and RBALoginModule [page 8].
- You have user and administrator permissions.
- You have configured the application to send passcodes via SMS. For more information, see Configuring Authentication with a Random Passcode Sent by SMS, E-Mail, or Another Channel [page 32].
- You have configured the user e-mail in the user management engine (UME). For more information, see Identity Management.
- You have created a *mail* policy script of type *Library* in the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts` with the following content:

```
var EMAIL = {
  send : function (recipient, subject, body, logger) {
    try {
      var server =
com.sap.security.api.UMFactory.getProperties().getDynamic("ume.notification.ma
il_host", "mail");
      var sender =
com.sap.security.api.UMFactory.getProperties().getDynamic("ume.notification.sy
stem_email", "no-reply@promptep.com");

      // Define the properties for the email and its gateway
      var properties = new java.util.Properties();
      properties.put("mail.smtp.host", server);
      // [uncomment to troubleshoot SMTP issues] properties.put("mail.debug",
true);

      var session = javax.mail.Session.getInstance(properties);
      // [uncomment to troubleshoot SMTP issues] session.setDebug(true);
      // [uncomment to troubleshoot SMTP issues]
session.setDebugOut(java.lang.System.err);
```

```
      var msg = new javax.mail.internet.MimeMessage(session);

      // Sender
      var addressFrom = new javax.mail.internet.InternetAddress(sender);
      msg.setFrom(addressFrom);

      // Recipient
      var addressTo =
java.lang.reflect.Array.newInstance(javax.mail.internet.InternetAddress, 1);
      addressTo[0] = new javax.mail.internet.InternetAddress(recipient);
      msg.setRecipients(javax.mail.Message.RecipientType.TO, addressTo[0]);

      // Set the header of the email
      msg.setSubject(subject, "utf-8");

      // Set the content of the email
      msg.setContent(body, "text/plain; charset=utf-8");

      // [uncomment to troubleshoot SMTP issues]
msg.writeTo(java.lang.System.err);

      // Send email
      javax.mail.Transport.send(msg);

      return true;
    } catch(err) {
      logger.traceError("Passcode cannot be sent via email.", err);
      return false;
    }
  }
}
```

For more information, see Working with Policy Scripts.

## Procedure

1. Log on to the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts`.
2. Create a new policy script with one of the scripts below and save your configurations. If you have created more than one version of your policy script, make sure that the version you want to be executed is activated. For more information on how to create a policy script, see Working with Policy Scripts.

   > → Recommendation
   >
   > To get a better grasp of the scenarios, you should try out the various scripts sequentially.

3. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
4. Choose the *Settings* tab.

   > i Note
   >
   > To allow applications to use your policy script, make sure that the *Policy* checkbox under the *Two-Factor Authentication* section is selected.

5. Choose the *Policy Script...* button under the *Two-Factor Authentication* section.
6. From the dropdown list select the policy script that you have just created, and reload it. Only the activated versions of policy scripts are visible in the dropdown list.

7. Save your configuration.
8. Test the scenario as a user.

### Related Information

Policy Scripts Implementation Guide
Working with Policy Scripts

## 1.3.3.1.1 Script 1: Control First Factor Logon

The following script sets conditions for authentication with Kerberos tokens or with a password.

```
function onInitialize(config, context) {
  if (context.getHttpClientContext().getClientIP() == "10.11.12.13") {
    config.setProperty("tfa.first.factor.login.module", "SPNegoLoginModule");
  } else {
    config.setProperty("tfa.first.factor.login.module",
"BasicPasswordLoginModule");
  }
}
```

### Related Information

Using Kerberos Authentication

## 1.3.3.1.2 Script 2: Set a Condition for Second Factor Logon

The following script controls the first factor logon and also requires a second factor for authentication when the user is a member of the *Managers* group.

```
function onInitialize(config, context) {
  if (context.getHttpClientContext().getClientIP() == "10.11.12.13") {
    config.setProperty("tfa.first.factor.login.module", "SPNegoLoginModule");
  } else {
    config.setProperty("tfa.first.factor.login.module",
"BasicPasswordLoginModule");
  }
}
function onFirstStageLogin(config, context, result) {
  var user = context.getLoginInfo().getUser();
  if (!user.isMemberOfGroup('GRUP.PRIVATE_DATASOURCE.un:Managers', true)) {
    result.doNotRequireSecondFactor();
  }
}
```

### Related Information

Assigning Principals to Roles or Groups

## 1.3.3.1.3 Script 3: Set Conditions for Out-of-Band Methods

This script accomplishes the following:

- Controls the first factor logon in accordance with a specific IP.
- Defines the second factor logon in accordance with the user role.
- Creates and sends a passcode via SMS if the user's OTP account is disabled.
- Creates and sends a passcode by e-mail if a mobile number is not specified in the user management engine.
  The script also defines the e-mail text and the message shown on the logon screen.
- Shows an error message and aborts the logon if the application cannot send a passcode.

```
#include "mail";
function onInitialize(config, context) {
  if (context.getHttpClientContext().getClientIP() == "10.11.12.13") {
    config.setProperty("tfa.first.factor.login.module", "SPNegoLoginModule");
  } else {
    config.setProperty("tfa.first.factor.login.module",
"BasicPasswordLoginModule");
  }
}
function onFirstStageLogin(config, context, result) {
  var logger = context.getLogger();
  var loginInfo = context.getLoginInfo();
  var user = loginInfo.getUser();
  if (!user.isMemberOfGroup('GRUP.PRIVATE_DATASOURCE.un:Managers', true)) {
    result.doNotRequireSecondFactor();
  }
  var totpInfo = loginInfo.getTOTPInfo();
  if (totpInfo.getStatus() == totpInfo.DISABLED || totpInfo.getStatus() ==
totpInfo.EXPIRED) {
    if (user.getCellPhone()) {
      config.setProperty("tfa.passcode.via.sms", "yes");
      var passcode = result.setRandomPasscode(10, 15, 5, "Passcode sent via SMS.
Please enter the passcode to log on.");
    } else if (user.getEmail()) {
      config.setProperty("tfa.passcode.via.sms", "no");
      var passcode = result.setRandomPasscode(10, 15, 5, "Passcode sent via
email. Please enter the passcode to log on.");
      EMAIL.send(user.getEmail(), "Access to CompanyA Inc. Portal", "Hello " +
user.getFirstName() + " " + user.getLastName() + ",\n\nTo log on, please enter
the following passcode: " + passcode + ".\n\nYour Portal IT Team", logger);
    } else {
      result.abortLogin('Passcode cannot be sent via SMS or email. Please
contact system administrator.');
    }
  }
}
```

## 1.3.3.1.4 Script 4: Set Time Constraints

This script accomplishes the following:

- Controls the first factor logon.
- Allows users to log on during working hours.
  The script defines working hours as Mon - Fri, from 9:00 to 18:00.
- Defines the second factor logon in accordance with the user role.
- Creates and sends a passcode via SMS or e-mail if a user does not have a supported device.
  The script also defines the e-mail text and the message shown on the logon screen.
- Shows an error message and aborts the logon if the application cannot send a passcode.

```
#include "mail";
function onInitialize(config, context) {
  if (context.getHttpClientContext().getClientIP() == "10.55.83.89") {
    config.setProperty("tfa.first.factor.login.module", "SPNegoLoginModule");
  } else {
    config.setProperty("tfa.first.factor.login.module",
"BasicPasswordLoginModule");
  }
}
function onFirstStageLogin(config, context, result) {
  var logger = context.getLogger();
  var loginInfo = context.getLoginInfo();
  var user = loginInfo.getUser();
  if (!isInWorkingHours(logger)) {
     result.abortLogin('You are not allowed to log on outside of working
hours.');
    return;
  }
  if (!user.isMemberOfGroup('GRUP.PRIVATE_DATASOURCE.un:Managers', true)) {
    result.doNotRequireSecondFactor();
  }
  var totpInfo = loginInfo.getTOTPInfo();
  if (totpInfo.getStatus() == totpInfo.DISABLED || totpInfo.getStatus() ==
totpInfo.EXPIRED) {
    if (user.getCellPhone()) {
      config.setProperty("tfa.passcode.via.sms", "yes");
      var passcode = result.setRandomPasscode(10, 15, 5, "Passcode sent via SMS.
Please enter the passcode to log on.");
    } else if (user.getEmail()) {
      config.setProperty("tfa.passcode.via.sms", "no");
      var passcode = result.setRandomPasscode(10, 15, 5, "Passcode sent via
email. Please enter the passcode to log on.");
      EMAIL.send(user.getEmail(), "Access to CompanyA Inc. Portal", "Hello " +
user.getFirstName() + " " + user.getLastName() + ",\n\nTo log on, please enter
the following passcode: " + passcode + ".\n\nYour Portal IT Team", logger);
    } else {
      result.abortLogin('Passcode cannot be sent via SMS or email. Please
contact system administrator.');
    }
  }
}
function isInWorkingHours(logger) {
  var now = new Date();
  logger.traceDebug("Login time " + now);
  return now.getDay() >= 1 && now.getDay() <= 5 && now.getHours() >= 9 &&
now.getHours() < 18;
}
```

## 1.3.3.2 Develop a Script for Context-Based Authorizations

### Prerequisites

- You have configured your policy configuration (application) to use the `TOTPLoginModule` with *otp&pwd* mode. For more information, see Configuring TOTPLoginModule and RBALoginModule [page 8].
- You have user and administrator permissions.
- You have configured the application to send passcodes via SMS. For more information, see Configuring Authentication with a Random Passcode Sent by SMS, E-Mail, or Another Channel [page 32].
- You have configured the user's e-mail in the user management engine (UME). For more information, see Identity Management.

### Procedure

1. Log on to the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts`.
2. Create a context-based script and save your configurations. If you have created more than one version of your policy script, make sure that the version you want to be executed is activated. For more information on how to create a policy script, see Working with Policy Scripts.

   This script accomplishes the following:

   - Controls first factor logon according to the IP range.
   - Decides on the second factor logon according to the first factor authentication method.
   - Does not require second factor for users that are not administrators.
   - Allows administrators to log on with full access permissions by providing a second factor. For the second factor logon, the application can send the password via SMS or email.

```
var EMAIL = {
  send : function (recipient, subject, message, logger) {
    try {
      var server =
com.sap.security.api.UMFactory.getProperties().getDynamic("ume.notification.ma
il_host", "mail");
      var sender =
com.sap.security.api.UMFactory.getProperties().getDynamic("ume.notification.sy
stem_email", "donotreply@acme.com");
      var props = new java.util.Properties();
      props.put("mail.smtp.host", server);
      var session = javax.mail.Session.getDefaultInstance(props, null);
      session.setDebug(false);
      var msg = new javax.mail.internet.MimeMessage(session);
      var addressFrom = new javax.mail.internet.InternetAddress(sender);
      msg.setFrom(addressFrom);
      var addressTo =
java.lang.reflect.Array.newInstance(javax.mail.internet.InternetAddress, 1);
      addressTo[0] = new javax.mail.internet.InternetAddress(recipient);
      msg.setRecipients(javax.mail.Message.RecipientType.TO, addressTo);
      msg.setSubject(subject, "utf-8");
      msg.setContent(message, "text/plain; charset=utf-8");
      javax.mail.Transport.send(msg);
      return true;
    } catch (err) {
      if (logger) {
```

```
        logger.traceError("Passcode cannot be sent via email.", err);
      }
      return false;
    }
  }
};
function onInitialize(config, context) {
  var httpClientContext = context.getHttpClientContext();
  if (httpClientContext.getClientIP().startsWith("10.11.12")) {
    config.setProperty("tfa.first.factor.login.module",
"ClientCertLoginModule, SPNegoLoginModule, BasicPasswordLoginModule");
  } else {
    config.setProperty("tfa.first.factor.login.module",
"ClientCertLoginModule, BasicPasswordLoginModule");
  }
}
function onFirstStageLogin(config, context, result) {
  var loginInfo = context.getLoginInfo();
  var authnMethod = loginInfo.getAuthenticationMethod();
  if (authnMethod == "certificate" || authnMethod == "spnego") {
    result.doNotRequireSecondFactor();
     return;
  }
  var user = loginInfo.getUser();
  if (user.isMemberOfGroup("GRUP.PRIVATE_DATASOURCE.un:Administrators", true)
||
      user.isMemberOfRole("ROLE.UME_ROLE_PERSISTENCE.un:Administrator",
true)) {
    result.setOptionalSecondFactor(
     "Select your access permissions to log on:",
     "Limited permissions",
     "Administrator permissions",
     result.DEFAULT_SELECTION_LIMITED_ACCESS);
  } else {
    result.doNotRequireSecondFactor();
  }
}
function onLimitedAccessSelected(config, context, result) {
  result.setAuthorizationScopes([], result.GRANT_ROLES_WITHOUT_SCOPES);
}
function onFullAccessSelected(config, context, result) {
  var logger = context.getLogger();
  var loginInfo = context.getLoginInfo();
  var user = loginInfo.getUser();
  var totpInfo = loginInfo.getTOTPInfo();
  if (totpInfo.getStatus() == totpInfo.DISABLED || totpInfo.getStatus() ==
totpInfo.EXPIRED) {
    if (user.getCellPhone()) {
      config.setProperty("tfa.passcode.via.sms", "yes");
      var passcode = result.setRandomPasscode(10, 15, 5, "Passcode sent via
SMS. Please enter the passcode to log on.");
    } else if (user.getEmail()) {
      config.setProperty("tfa.passcode.via.sms", "no");
      var passcode = result.setRandomPasscode(10, 15, 5, "Passcode sent via
email. Please enter the passcode to log on.");
      EMAIL.send(user.getEmail(), "Access to CompanyA Inc. Portal", "Hello "
+ user.getFirstName() + " " + user.getLastName() + ",\n\nTo log on, please
enter the following passcode: " + passcode + ".\n\nYour Portal IT Team",
logger);
    } else {
      result.abortLogin('Passcode cannot be sent via SMS or email. Please
contact system administrator.');
    }
  }
}
function onSecondStageLogin(config, context, result) {
  result.setAuthorizationScopes(["FULL_ACCESS"],
result.GRANT_ROLES_WITHOUT_SCOPES);
```

```
}
```

3. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
4. Choose the *Settings* tab.

> **i Note**
>
> To allow applications to use your policy script, make sure that the *Policy* checkbox under the *Two-Factor Authentication* section is selected.

5. Choose the *Policy Script...* button under the *Two-Factor Authentication* section.
6. From the dropdown list select the policy script that you have just created, and reload it. Only the activated versions of policy scripts are visible in the dropdown list.
7. Save your configuration.
8. Test the scenario as a user.

### Related Information

Working with Policy Scripts

## 1.3.3.3 Develop a Script for External Passcode Validation

By implementing scripts, you can use external passcode validation services. For example you can use RSA SecureID passcode to log on to applications.

### Prerequisites

- You have configured your policy configuration (application) to use the `TOTPLoginModule` with *otp&pwd* mode. For more information, see Configuring TOTPLoginModule and RBALoginModule [page 8].
- You have administrator and user permissions.
- You have configured a `RADIUS` destination in the Secure Login Administration Console. For more information, see Managing Destinations.
- You have configured your server for authentication with RSA passcodes. You also have to configure the users who can use this authentication.
- You have created the users in the AS Java user management engine (UME). For more information about how to add those users, see Creating a Technical User.
- You have created an *RSA_Users* group in the UME and have assigned your users to the group. For more information, see Assigning Principals to Roles or Groups.
- You have created a policy script named *SAP* of type *Library* in the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts` with the following content:

```
/**
```

```
* This library defines the following namespaces: SAP, SAP.sample and SAP.util.
* It has to be included before any other library provided by SAP.
*/
var SAP = {
    sample : {},
    util : {}
}
```

- You have created a policy script named *SAP_util_rsa* of type *Library* in the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts` with the following content:

```
/**
* This library provides an utility class for passcode validation against RSA
SecurID server.
* For additional information check the online documentation at http://
help.sap.com/saphelp_nwsso20/helpdata/en/2c/c4781832aa4cd9bf5a253aa0ac1445/
content.htm
*/
SAP.util.rsa = {
    PasscodeValidator : {
        /**
         * Function to validate RSA passcode against a RSA server.
         * Supports RSA server failover via multiple RADIUS destinations
         *
         *
         * @param  {PolicyConfiguration} config  the configuration used for
the current login. Different configuration options can be read and set
         * @param  {PolicyContext} context  policy context for the current
login
         * @param  {PasscodeValidationLoginResult} result passcode validation
result. End-user messages (warnings, errors, infos) are returned here along
with the passcode validation status
         * @param  {String} username username used in the current login
         * @param  {String} passcode passcode to be validated
         * @param  {String[]} [destinations] array with the names of RADIUS
destinations for failover support.
         *          When it is null or empty, the RADIUS destination name is
taken from the login module option
"rsa.login.module.option.RadiusDestination".
         *          When there is no such login module option the default
RADIUS destination is used by "SecureLoginModule20RADIUS" login module.
         * @return {boolean} true when the given passcode is valid, false
when is not valid or the validation is not possible
         */
        validate: function (config, context, result, username, passcode,
destinations) {
            // The type of "logger" is LogAccessor
            var logger = context.getLogger();
            logger.traceDebug("Validating passcode against RSA server for
user: " + username);
            // The type of "rsaContext" is RSAContext
           var rsaContext = context.getRsaContext();
            // RSA context is using the login module configured with
configuration property "rsa.login.module" and default value
"SecureLoginModule20RADIUS" for passcode validation.
            // "passcodeStatus" is of type String with values
"VALID_PASSCODE", "INVALID_PASSCODE", "VALID_NEXT_PASSCODE_REQUIRED" or
"RSA_SERVICE_UNAVAILABLE".
            var passcodeStatus;
            var processedPasscodeStatus;
            // no failover
            if (typeof destinations == "undefined" || destinations == null ||
destinations.length == 0) {
                logger.traceInfo("No failover RADIUS destinations are
specified.");
                passcodeStatus = rsaContext.validatePasscode(username,
passcode);
```

```
                processedPasscodeStatus =
processPasscodeStatus(passcodeStatus);
                if (processedPasscodeStatus == null) {
                    logger.logError("The RADIUS destination was not available
for passcode validation.");
                    result.setValidationFailureWithErrorMessage("Unable to
validate RSA SecurID passcode. Contact your system administrator.");
                    return false;
                }
                return processedPasscodeStatus;
            }
            // failover
            logger.traceInfo("Using the following RADIUS destinations: " +
destinations);
            var destinationsCount = destinations.length;
            for (var i = 0; i < destinationsCount; i++) {
                if (!destinations[i]) {
                    continue;
                }
                rsaContext.setRADIUSDestination(destinations[i]);
                passcodeStatus = rsaContext.validatePasscode(username,
passcode);
                processedPasscodeStatus =
processPasscodeStatus(passcodeStatus);
                if (processedPasscodeStatus == null) {
                    continue;
                }
                return processedPasscodeStatus;
            }
            // unsuccessful failover
            logger.logError("None of the configured RADIUS destinations were
available for passcode validation.");
            result.setValidationFailureWithErrorMessage("Unable to validate
RSA SecurID passcode. Contact your system administrator.");
            return false;
            function processPasscodeStatus(passcodeStatus) {
                switch (passcodeStatus) {
                    // the configured RADIUS server is not available
                    case rsaContext.RSA_SERVICE_UNAVAILABLE: {
                        return null;
                    }
                    // Passcode validation passed successfully.
                    case rsaContext.VALID_PASSCODE:
                    {
                        logger.traceDebug("RSA SecurID passcode is
accepted.");
                        result.setValidationSuccessful();
                        return true;
                    }
                    // Passcode is invalid. Issue a warning message.
                    case rsaContext.INVALID_PASSCODE:
                    {
                        logger.traceDebug("Invalid RSA SecurID passcode");
                        // A warning message is displayed on the logon page.
                        result.setValidationFailureWithWarningMessage("Wrong
RSA SecurID passcode");
                        return false;
                    }
                    // Provide the next passcode. For example, when several
wrong passcodes were entered on previous attempts.
                    case rsaContext.VALID_NEXT_PASSCODE_REQUIRED:
                    {
                        logger.traceDebug("RSA SecurID passcode is accepted.
User should provide next one for confirmation");
                        // An information message is displayed on the logon
page in case a second passcode is required.
                        result.setValidationFailureWithInfoMessage("RSA
SecurID passcode accepted. Enter a second passcode to log on.");
```

```
                        return true;
                    }
                    default:
                    {
                        logger.traceDebug("Unknown RSA SecurID passcode
status: " + passcodeStatus);
                        result.setValidationFailureWithErrorMessage("Unknown
RSA SecurID passcode status. Contact your system administrator.");
                        return false;
                    }
                }
            }
        }
    }
};
```

## Procedure

1. Log on to the *Policy Script Administration Console* at `http(s)://<host>:<port>/ssoadmin/scripts`.
2. Create the passcode validation script and save your configurations. If you have created more than one version of your policy script, make sure that the version you want to be executed during authentication is activated. For more information on how to create a policy script, see Working with Policy Scripts.
3. Log on to the *One-Time Password Administration* UI at `http(s)://<host>:<port>/ssoadmin/otp`.
4. Choose the *Settings* tab.

> **i** Note
>
> To allow applications to use your policy script, make sure that the *Policy* checkbox under the *Two-Factor Authentication* section is selected.

5. Choose the *Policy Script...* button under the *Two-Factor Authentication* section.
6. From the dropdown list select the policy script that you have just created, and reload it. Only the activated versions of policy scripts are visible in the dropdown list.
7. Save your configuration.
8. Test the scenario as a user.

## Related Information

Working with Policy Scripts

# 1.3.3.3.1 Script 1: Using RSA SecurID as External Passcode Validation Service

This script performs the following tasks:

- Sets a property that allows external passcode validation if the user is a member of a specific group.
- Validates the external passcode that the user provides and informs the user about the log-on result.

```
#include SAP;
#include SAP_util_rsa;
function onFirstStageLogin(config, context, result){
    var loginInfo = context.getLoginInfo();
    if
(loginInfo.getUser().isMemberOfGroup("GRUP.PRIVATE_DATASOURCE.un:RSA_Users",
true)) {
        config.setProperty("otp.use.external.passcode.validation", "yes");
    }
}
function validatePasscode(config, context, result, username, passcode) {
    SAP.util.rsa.PasscodeValidator.validate(config, context, result, username,
passcode);
}
```

#### 1.3.3.3.2 Script 2: Supporting Server Failover When Using RSA SecurID as External Passcode Validation Service

This script performs the following tasks:

- Sets a property that allows external passcode validation if the user is a member of a specific group.
- Tries out the set RADIUS destinations until one validates the passcode and informs the user about the log-on result.

```
#include SAP;
#include SAP_util_rsa;
function onFirstStageLogin(config, context, result){
    var loginInfo = context.getLoginInfo();
    if
(loginInfo.getUser().isMemberOfGroup("GRUP.PRIVATE_DATASOURCE.un:RSA_Users",
true)) {
        config.setProperty("otp.use.external.passcode.validation", "yes");
    }
}
function validatePasscode(config, context, result, username, passcode) {
    var destinations = ["RADIUSDestination1", "RADIUSDestination2",
"RADIUSDestination3"];
    SAP.util.rsa.PasscodeValidator.validate(config, context, result, username,
passcode, destinations);
}
```

## 1.4 One-Time Password Authentication User Guide

This document provides information about how users can set up an account for their authenticator mobile application.

The authenticator is a mobile application that you install on your mobile devices to generate passcodes. In order to use these passcodes, you have to add an account on your mobile devices.

> **i** Note
>
> The OTP setup is compatible with third-party authenticators such as Google Authenticator or Microsoft Authenticator. If you want to use all the features of OTP authentication however, such as online setup or mobile SSO, you need SAP Authenticator.

### Related Information

SAP Authenticator Mobile Application [page 95]
Setting Up an Authenticator Account on Your Mobile Device [page 96]
Disabling an Authenticator Account on Your Mobile Devices [page 98]
Logging On to Different Systems with One-Time Passwords [page 99]

## 1.4.1 SAP Authenticator Mobile Application

SAP Authenticator is a mobile application that generates passcodes for systems that require one-time password authentication. Passcodes are time-based and valid for one logon attempt, meaning they are more secure than common static passwords.

You can use SAP Authenticator on your mobile device after completing the configuration in the *Mobile Device Setup* UI. Provided you have a user permission, you can install SAP Authenticator and set up an account on multiple mobile devices. The application runs on iOS and Android mobile operating systems.

You can set up your account in one of the following ways:

- By regular setup
  You add an account and then set the account's applications for mobile SSO manually.
  You add an application by scanning a QR code or by entering the application's URL. The QR code can be displayed on the application's logon page. If it is not, this information should be provided by your administrator. After adding the applications, you can log on to them from SAP Authenticator without having to enter a username and passcode (through mobile SSO).
  You can also access an account's application by choosing the link for logging on with SAP Authenticator on your mobile device's browser.
- By online setup
  You add your account by providing a confirmation code. All applications for mobile SSO are included in the account automatically.

You can also use the *Update* button in SAP Authenticator to include any recent changes of the applications for your account.

Once you have finished the setup process for SAP Authenticator, your mobile device is linked to your user account. From then on, you can use the passcodes generated by the mobile application for the setup account, in order to log on to any systems or applications that require one-time password authentication.

> **i Note**
>
> Depending on your administrator's configuration, some logon applications might require you to enter the passcode and password in one field, either concatenated or with a separator.

When you log on to an application, you might see a checkbox with *Trust this device* (when using a device) or *Trust this computer* (when using a computer). You have to provide the two factors when you enable this feature. After the feature is enabled, you log on with one credential only (a passcode or a password for example).

### Related Information

Setting Up an Authenticator Account on Your Mobile Device [page 96]
Disabling an Authenticator Account on Your Mobile Devices [page 98]
Logging On to Different Systems with One-Time Passwords [page 99]

## 1.4.2 Setting Up an Authenticator Account on Your Mobile Device

To use an authenticator application on your mobile device, you have to set up an account. You can set up the account on multiple mobile devices.

> **i Note**
>
> Due to your user account having a limited validity period, you have to disable the account and set it up again on all your mobile devices when it is about to expire.
>
> Depending on the validity and expiration period set in the *One-Time Password Administration* UI, following operations are allowed:

| Secret Key Validity | Can use the passcode for authentication? | Can register new device on the *Mobile Device Setup* page? | Can disable device on the *Mobile Device Setup* page? |
|---|---|---|---|
| *Expiration Warning Period* is not reached | Yes | Yes | Yes |

| Secret Key Validity | Can use the passcode for authentication? | Can register new device on the *Mobile Device Setup* page? | Can disable device on the *Mobile Device Setup* page? |
|---|---|---|---|
| During *Expiration Warning Period* | Yes<br>Warning is shown | No<br>You can disable the registered device and then create a new registration. | Yes |
| Secret key has expired | No<br>Error message is shown that the registration has expired | No<br>You can disable the registered device and then create a new registration. | Yes<br>For security reasons, passcode of expired device is required |

## Context

You can set up the account as follows, depending on your administrator's system configuration:

- By regular setup
  For this setup, you can use any compatible authenticator application. You perform the setup by scanning a QR code or entering a secret key.
- By online account setup
  For this setup, you can use only SAP Authenticator, which is available for iOS and Android devices. You perform this setup by scanning a QR code or entering a setup URL and by providing a confirmation code.

## Procedure

1. Log on to the *Mobile Device Setup* UI at `http(s)://<host>:<port>/otp`.

   > **i Note**
   >
   > This is a self-service application that your administrator should provide to you by e-mail or as a link.

2. Install the authenticator application on your mobile device.

   To use SAP Authenticator, install it by scanning a QR code (if your administrator has configured an installation section in the *Mobile Device Setup* UI) or from iTunes (for iOS devices) or Google Play (for Android devices).

   Once you have installed SAP Authenticator, an icon is created on your mobile device screen.

3. Choose *Set Up Account on Device* in the *Mobile Device Setup* UI.

   This operation opens a page with instructions how to set up the account on your device.

> **i Note**
>
> If you have already set up the account on another device, the application will prompt you to enter a passcode generated by the device.

4. Follow the instructions on the page.

   Once the setup has been completed, you receive the message `Account setup completed` at the top of the main page.

> **i Note**
>
> If you use SAP Authenticator, the mobile app starts generating a passcode every 30 seconds.

### Related Information

SAP Authenticator Mobile Application [page 95]
Disabling an Authenticator Account on Your Mobile Devices [page 98]
Logging On to Different Systems with One-Time Passwords [page 99]

## 1.4.3 Disabling an Authenticator Account on Your Mobile Devices

### Context

You can disable the authenticator account on your mobile devices in one of the following ways:

- By contacting your system administrator.
  You need to contact your system administrator if any of your devices are lost or stolen for example.
- By completing the steps in the *Mobile Device Setup* UI.
  You need at least one device containing this account so that you can provide a valid passcode.

### Procedure

1. Log on to the *Mobile Device Setup* UI at `http(s)://<host>:<port>/otp`.

> **i Note**
>
> This is a self-service application that your administrator should provide to you by e-mail or as a link.

2. Choose the *Disable Account* link.

   Another page opens,providing instructions about how to disable your account.
3. Enter a passcode for this account and choose the *Disable Account* button.

   If the process is successful, you receive the message `Account disabled; delete the account from all devices` at the top of the main page.
4. Delete the account on all devices.

   You have to do this manually because the account is not automatically removed from your devices.

### Related Information

SAP Authenticator Mobile Application [page 95]
Setting Up an Authenticator Account on Your Mobile Device [page 96]
Logging On to Different Systems with One-Time Passwords [page 99]

## 1.4.4 Logging On to Different Systems with One-Time Passwords

You can log on to systems that require one-time password authentication with passcodes generated by an authenticator mobile application in a number of different ways:

- Log on to systems using Secure Login Client
- Log on to systems using SAML, using Application Server Java as an identity provider (IdP)
- Log on to web applications running on SAP NetWeaver AS for Java

Depending on the administrator's configuration, you need to provide passcodes either for single-factor authentication or for two-factor authentication. The following options are available:

- Single-factor authentication is required.
  You log on to applications with a single factor, either a passcode or another credential. This option can also be configured to require two passcodes for higher security.
- Two-factor authentication is required.
  You can log on to a system with two means of identification: a user name and a passcode, and another factor for authentication (for example, a corporate password, X.509 certificate or others). This option can be configured to require a single factor after the first successful logon.
- The required factors are determined at runtime.
  An application can decide dynamically whether to require a single factor or two factors of authentication.

### Related Information

SAP Authenticator Mobile Application [page 95]

Setting Up an Authenticator Account on Your Mobile Device [page 96]
Disabling an Authenticator Account on Your Mobile Devices [page 98]

## 1.4.5 User Error Messages

This document explains some of the error messages that users receive while using OTP-related applications.

### Setup Account Messages

You could receive any of the following messages when setting up an authenticator account on your mobile device:

| Error Message | Description | Solution |
|---|---|---|
| `Wrong passcode; enter passcode again` | You have entered a wrong passcode when setting up an account on additional device or when disabling an account. | Reenter the passcode generated by your mobile device.<br>**i Note**<br>Once entered, a passcode cannot be used a second time. If you don't want to wait for the next passcode to appear on your SAP Authenticator screen, you can swipe the *Current Passcode* screen in order to see the next passcode. |
| | You have not entered the passcode within the specified time interval. | Enter the passcode within the specified time interval. It is 30 seconds for users with the SAP Authenticator installed. |
| | Your clock is not synchronized with the clock on the server. | Configure your mobile device to automatically update the date and time under *Settings* > *General* > *Date & Time* > *Set Automatically* or contact your administrator to check the time on the server. |
| `Cannot generate QR code` | A QR code cannot be generated for the setup process. | Restart the account setup on the *Mobile Device Setup* UI. |

| Error Message | Description | Solution |
| --- | --- | --- |
| `Custom alias not found: <alias>; contact your system administrator` | The URL path for the custom installation section is not correct. | Contact your system administrator to check the configured alias for the custom installation section of the *Mobile Device Setup* UI. |
| `Error while connecting to alias <alias>; contact system administrator` | The application cannot create the custom installation section from the configured URL. | Contact your system administrator to check the configured alias for the custom installation section of the *Mobile Device Setup* UI. |
| `Error occurred` | An unspecified error has occurred. | Restart the account setup on the *Mobile Device Setup* UI.<br>Refresh your browser.<br>Contact your system administrator. |

## Logon Messages

You receive some of the following messages when you log on to applications:

| Error Message | Description | Solution |
| --- | --- | --- |
| `User authentication failed` or `Authentication failed` | You have entered an invalid password or passcode. | • Make sure that you have enabled automatic time updates on your mobile device (usually under *Settings* > *Date and Time* > *Automatic*).<br>• Try again or contact your administrator to check that your mobile device's time is synchronized with the time of the OTP server. |
| | You have entered a valid password and passcode, but the username is wrong. | Ask the administrator to check if the **UserMappingMode** option is configured to require authentication with logon ID or email. |
| `An authentication problem occurred; contact your system administrator` | This error appears when the administrator has configured a wrong policy script. | Contact your system administrator to fix the policy script in the *One-Time Password Administration* UI. |

| Error Message | Description | Solution |
| --- | --- | --- |
| `Authentication failed; password locked` | You have entered valid credentials while your passcode was locked. | • You can wait until your passcode is unlocked according to the specified unlock period. You can then log on again.<br>• You can contact your system administrator to unlock your account. |
| `Logon with a passcode is required. For the generation of passcodes, a mobile device has to be activated.` | You need to provide a one-time password (passcode) in order to log on to the application. | • Set up an authenticator account on your device in the *Mobile Device Setup* UI.<br>• If you cannot install an authenticator application on your mobile device, contact your system administrator for alternative solutions. |
| `Possible session fixation attack detected; contact your system administrator` | The application has detected vulnerability in your system. | Please contact your administrator to find out what is causing this problem and then try to log on again. |
| `Possible logon XSRF attack is detected. Please contact your system administrator.` | The application has detected vulnerability in your system. | Please contact your administrator to find out what is causing this problem and then try to log on again. |
| `Cannot send passcode by SMS; contact your system administrator` | The issue might occur because of the following:<br>• You have changed the phone number of your mobile device.<br>• You have changed the provider of your mobile device.<br>• Other configuration issues. | Please contact your system administrator to find out what is causing this problem. |
| `Wrong passcode` | You have entered the wrong passcode during logon. | Try again.<br>**i Note**<br>Once entered, a passcode cannot be used a second time. If you don't want to wait for the next passcode to appear on your SAP Authenticator screen, you can swipe the *Current Passcode* screen in order to see the next passcode. |

| Error Message | Description | Solution |
| --- | --- | --- |
| | You have not entered the passcode within the specified time interval. | Enter the passcode within the specified time interval. This is 30 seconds for users with the SAP Authenticator application installed. |
| | Your clock is not synchronized with the clock on the server. | Configure your mobile device to automatically update the date and time under *Settings* > *General* > *Date & Time* > *Set Automatically* or contact your administrator to check the time on the server.. |

### Related Information

One-Time Password Authentication User Guide [page 95]

## 1.5 One-Time Password Authentication Security Guide

This guide provides recommendations how administrators and developers can secure applications that use one-time password authentication.

### Before You Start

One-time password (OTP) authentication allows you to log on to systems using Secure Login Client, or using identity provider or web applications running on AS Java. Before you configure OTP authentication with any of those systems, you should familiarize yourself with the security requirements of the system you are using in your landscape.

- For security information about a system running on AS Java, see SAP NetWeaver Application Server Java Security Guide
- For security information about a system using Secure Login, see Secure Login Security Guide

### User Administration and Authentication

With the *One-Time Password Administration* UI, you can manage user accounts in order to solve various security issues. For more information, see Managing User Accounts [page 43].

When you configure user authentication, you should consider the following security recommendations:

- Configure two-factor authentication if your scenario allows it.
  We recommend using two-factor authentication when possible, as this requires the user to provide two authentication factors: something the user knows (a password for example), and something the user has (a mobile device that generates a passcode). For more information, see Configuring Two-Factor Authentication [page 30].
- If you use single-factor authentication, configure the application to require two distinct passcodes.
  We recommend this option as the best protection against malicious attacks. For more information, see Configuring Single-Factor Authentication [page 28].
- If you use single-factor authentication, configure the application to require user confirmation during automatic logon.
  We recommend this configuration for mobile single sign-on scenarios where this kind of user confirmation can protect the application from an XSRF logon attack or from a man-in-the-middle attack. For more information, see Configuring Single-Factor Authentication [page 28].
- Configure the OTP-related logon application if your scenario allows it.
  We recommend the OTP-related logon application, as it protects against XSRF and session fixation attacks. For more information, see Configuring an OTP-Related Logon Application [page 38].

## Authorizations

To use OTP-related tools, users and administrators need specific roles assigned in SAP NetWeaver Administrator. We recommend only assigning these roles to the required groups, roles, or users. These assignments are not configured by default. For more information, see One-Time Password Authentication Administration Guide [page 5].

## Passcode Security

- Digest Algorithm
  We recommend using the most secure password generation algorithm for your scenario, such as SHA-512. For more information, see Additional Settings [page 51].
- Passcode length
  We recommend using 8-digit passcodes. For more information, see Additional Settings [page 51].

## Account Protection

- Account locking
  Use a minimum number of usable logon attempts before an account is locked in order to balance security and usability. For more information, see Additional Settings [page 51].
- Secret key expiration period
  Set a reasonable expiration period for the secret key. We recommend 365 days. For more information, see Additional Settings [page 51].

## Session Security

- *Remember client (persistent cookie)* option
  If you enable this option in the *One-Time Password Administration* UI, we recommend using the default *HTTP only* and *Secure* attributes for the cookie.
- XSRF protection
  XSRF protection is enabled by default for the *TOTPLoginModule*. We recommend not disabling it with the *tfa.enable.xsrf.protection* option. For more information, see One-Time Password Login Module Options [page 13].

## Network and Communication Security

We recommend configuring the system using OTP authentication to be accessed through *HTTPS*.

## Security of the Mobile Application

We recommend that users protect the SAP Authenticator mobile application with a logon password.

## Security-Relevant Logging and Tracing

- If your system uses AS Java, check the security logs and audit logs from SAP NetWeaver Administrator. For more information, see Logging and Tracing.
- If your system uses Secure Login Client, check the client trace. For more information, see Tracing Secure Login Client.

## Related Information

SAP NetWeaver Security Guide
Security on SAP Community Network
SAP Security Notes
Security at SAP

# 1.6 Troubleshooting

This section can help you solve issues with the configuration or usage of one-time password authentication.

It gives information about the error messages users receive and how administrators can view and collect logs and traces.

## Support

If you cannot solve your issues with the provided information, you can create an *Incident* on SAP Support Portal with a component `BC-IAM-SSO-OTP`.

## Related Information

User Error Messages [page 107]
SAML 2.0 Authentication – Limitations [page 110]
Collecting Traces with the Security Troubleshooting Wizard [page 111]

## 1.6.1 User Error Messages

This document explains some of the error messages that users receive while using OTP-related applications.

### Setup Account Messages

You could receive any of the following messages when setting up an authenticator account on your mobile device:

| Error Message | Description | Solution |
| --- | --- | --- |
| `Wrong passcode; enter passcode again` | You have entered a wrong passcode when setting up an account on additional device or when disabling an account. | Reenter the passcode generated by your mobile device.<br>**i Note**<br>Once entered, a passcode cannot be used a second time. If you don't want to wait for the next passcode to appear on your SAP Authenticator screen, you can swipe the *Current Passcode* screen in order to see the next passcode. |
| | You have not entered the passcode within the specified time interval. | Enter the passcode within the specified time interval. It is 30 seconds for users with the SAP Authenticator installed. |
| | Your clock is not synchronized with the clock on the server. | Configure your mobile device to automatically update the date and time under *Settings* > *General* > *Date & Time* > *Set Automatically* or contact your administrator to check the time on the server. |
| `Cannot generate QR code` | A QR code cannot be generated for the setup process. | Restart the account setup on the *Mobile Device Setup* UI. |
| `Custom alias not found: <alias>; contact your system administrator` | The URL path for the custom installation section is not correct. | Contact your system administrator to check the configured alias for the custom installation section of the *Mobile Device Setup* UI. |

| Error Message | Description | Solution |
| --- | --- | --- |
| `Error while connecting to alias <alias>; contact system administrator` | The application cannot create the custom installation section from the configured URL. | Contact your system administrator to check the configured alias for the custom installation section of the *Mobile Device Setup* UI. |
| `Error occurred` | An unspecified error has occurred. | Restart the account setup on the *Mobile Device Setup* UI. |
| | | Refresh your browser. |
| | | Contact your system administrator. |

## Logon Messages

You receive some of the following messages when you log on to applications:

| Error Message | Description | Solution |
| --- | --- | --- |
| `User authentication failed` or `Authentication failed` | You have entered an invalid password or passcode. | • Make sure that you have enabled automatic time updates on your mobile device (usually under *Settings* > *Date and Time* > *Automatic*).<br>• Try again or contact your administrator to check that your mobile device's time is synchronized with the time of the OTP server. |
| | You have entered a valid password and passcode, but the username is wrong. | Ask the administrator to check if the **UserMappingMode** option is configured to require authentication with logon ID or email. |
| `An authentication problem occurred; contact your system administrator` | This error appears when the administrator has configured a wrong policy script. | Contact your system administrator to fix the policy script in the *One-Time Password Administration* UI. |
| `Authentication failed; password locked` | You have entered valid credentials while your passcode was locked. | • You can wait until your passcode is unlocked according to the specified unlock period. You can then log on again.<br>• You can contact your system administrator to unlock your account. |

| Error Message | Description | Solution |
| --- | --- | --- |
| `Logon with a passcode is required. For the generation of passcodes, a mobile device has to be activated.` | You need to provide a one-time password (passcode) in order to log on to the application. | • Set up an authenticator account on your device in the *Mobile Device Setup* UI.<br>• If you cannot install an authenticator application on your mobile device, contact your system administrator for alternative solutions. |
| `Possible session fixation attack detected; contact your system administrator` | The application has detected vulnerability in your system. | Please contact your administrator to find out what is causing this problem and then try to log on again. |
| `Possible logon XSRF attack is detected. Please contact your system administrator.` | The application has detected vulnerability in your system. | Please contact your administrator to find out what is causing this problem and then try to log on again. |
| `Cannot send passcode by SMS; contact your system administrator` | The issue might occur because of the following:<br>• You have changed the phone number of your mobile device.<br>• You have changed the provider of your mobile device.<br>• Other configuration issues. | Please contact your system administrator to find out what is causing this problem. |
| `Wrong passcode` | You have entered the wrong passcode during logon. | Try again.<br>**i Note**<br>Once entered, a passcode cannot be used a second time. If you don't want to wait for the next passcode to appear on your SAP Authenticator screen, you can swipe the *Current Passcode* screen in order to see the next passcode. |
| | You have not entered the passcode within the specified time interval. | Enter the passcode within the specified time interval. This is 30 seconds for users with the SAP Authenticator application installed. |

| Error Message | Description | Solution |
| --- | --- | --- |
| | Your clock is not synchronized with the clock on the server. | Configure your mobile device to automatically update the date and time under *Settings* > *General* > *Date & Time* > *Set Automatically* or contact your administrator to check the time on the server.. |

### Related Information

One-Time Password Authentication User Guide [page 95]

## 1.6.2 SAML 2.0 Authentication – Limitations

If you try to authenticate to a business application by using both SAML 2.0 authentication and the *SAP Authenticator* mobile app, you will not succeed.

Below are explained use cases and the reasons.

### SSO is initiated by the service provider

You use two-factor authentication with Single Sign-On and *SAP Authenticator*, where the SSO is initiated by a SAML 2.0 service provider (SP). Since *SAP Authenticator* can trigger authentication only by a URL, the authentication won't succeed.

As a result, when you click the *Log-on with SAP Authenticator* link, you experience a never-ending authentication loop.

### SSO is initiated by the identity provider

You use two-factor authentication with Single Sign-On and *SAP Authenticator*, where the SSO is initiated by an identity provider (IdP). The SP is requesting a forced re-authentication while, at the same time, an existing IdP session is still active.

As a result, the forced re-authentication triggers an SP-initiated SSO **after** receiving the SAML message for successful IdP login. This ends up in error **403 Forbidden**.

## Related Information

Configuring AS Java as a Service Provider

Configuring AS ABAP as a Service Provider

Trusting an Identity Provider

# 1.6.3 Collecting Traces with the Security Troubleshooting Wizard

## Context

You can use the *Security Troubleshooting Wizard* plug-in of SAP NetWeaver Administrator to collect traces for various OTP-related issues.

## Procedure

1. Open the Troubleshooting Wizard.
2. Choose the *Manage Incidents* link.
3. Choose the *Copy to New* button and copy the *Authentication* incident.
4. And add the following trace locations:
   - `com.sap.security.sso.authlib`
   - `com.sap.security.accesspolicies`
   - `com.sap.security.jaas.otp`
   - `com.sap.security.sso`
   - `com.sap.securelogin`

   > **i Note**
   >
   > You need the `com.sap.security.sso.authlib` trace location to get OTP-related logs and traces.
5. Collect the traces for the incident.

## Related Information

Collecting Traces for Troubleshooting Security Problems

User Error Messages [page 107]

Managing Incidents

Log Viewer

# Important Disclaimers and Legal Information

## Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

## Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

## Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

## Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

## Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap



THE BEST RUN SAP