Transition	Position
pwreset	1
pwreset,mfa	2

UserInformation_messageType

- `error` - terminates the session (`AUTH_ERROR`) and shows an error message.
- `warn` - does not terminate the session (`AUTH_CONTINUE`) but the message is shown as an error.
- `info` - renders as a message of type `info` and does not terminate the session (`AUTH_CONTINUE`).

Terminating the session needs careful testing as state loss can lead to follow-up errors.

NevisAdaptDatabase_oracleDataTablespaceName

Name of the data tablespace for the oracle database used for the Kubernetes migration. It's recommended to keep the default value unless the pattern is used with an existing database that has a different one.

NevisAuthRealmBase_auth

Assign a `nevisAuth Instance` pattern.

JWTToken_keystore

A Key Store is required when an asymmetric algorithm is used.

This is required for `JWE` because of the `RSA-OAEP-256` algorithm.

NevisFIDODeployable_registrationTimeout

Defines the maximum time duration between the generation of the `RegistrationRequest` by nevisFIDO and the `RegistrationResponse` by the FIDO UAF client.

If the client has not sent the response after this time, a client timeout occurs.

The default value is 5 minutes. If no time unit is provided, seconds will be used.

This timeout is relevant in registration use-cases, such as:

- In-Band Registration
- Out-of-Band Registration

AccessRestriction_ipHeader

Optional setting used to specify HTTP header that contains the users IP. Otherwise, a default environment variable from nevisProxy is used.

Examples:

- `X-Forwarded-For`

NevisAuthDeployable_server

Set *Server Configuration* properties (`nevisauth.yml`).

Examples:

```
server.max-http-header-size: 16384
```

SamlIdpConnector_signatureValidation

Configure for which SAML elements signature validation shall be performed.

It is recommended that the IDP signs the entire `Response` as this is the most secure mode.

If only the `Assertion` is signed nevisAuth will perform additional checks to prevent attack scenarios.

NevisMetaDeployable_logging

Configure the nevisMeta log files.

GenericThirdPartyRealm_authService

Optionally assign an application which provides the authentication service and shall be exposed on the same virtual host as the applications.

Not required for federation-based authentication where the authentication service is hosted on another domain.

KeyObject_keyStoreName

Define the `name` of the parent `KeyStore` element.

You can enter the `name` of a `KeyStore` element generated by another pattern, or enter a new name.

If **not** configured, the `name` of the `KeyStore` depends on **where** this `nevisAuth KeyObject` pattern is assigned:

- `Generic Authentication Realm` : sanitized name of the realm pattern.
- `Generic Authentication Step` : sanitized name of the step pattern.
- `nevisAuth Instance` : the name `AddonKeyStore` .

Note that the nevisAuth configuration always contains a `KeyStore` element with name `DefaultKeyStore` . This `KeyStore` is typically used as a container for signer key material.

NevisDPDeployable_resources

Upload additional resources here.

For instance, you may upload files used by your data sources in the `Configuration` .

nevisAdmin generation time expressions (e.g. `${var.name}`) are not supported here.

SamlToken_audience

Configure the `AudienceRestriction` .

Enter 1 line for each `Audience` .

AppleLogin_clientSecret

The Client Secret is a JWT token generated by using a private key provided by Apple. Please follow the instructions here.

You can generate the client secret by yourself and configure it here, or upload the Private Key to generate the client secret automatically. Only private key or client secret can be use at the time

NevisIDMCheckUserCredentials_anyCredentialFound

Configure the step to execute if the user has at least one credential from credential type selected in `Credential Types` , but nit from all credential type. If no step is configured here the process ends with `AUTH_DONE` .

NevisAuthDeployable_signerKeyStore

Assign a key store for signing the **internal** NEVIS SecToken.

This token is returned to nevisProxy and validated there. It should **not** be added to calls to applications. If your applications need a NEVIS SecToken, assign a `NEVIS SecToken` pattern to your applications.

If no pattern is assigned, the signer key material will be provided by automatic key management.

NevisIDMDeployable_clientAuth

Setting for 2-way TLS on the nevisIDM HTTPs endpoint. There are 3 options will affect the callers (e.g. nevisProxy or technical clients accessing nevisIDM REST APIs)

- required: Callers **must** present a client certificate.
- requested: Callers **can** present a client certificate.
- disabled: Callers **must not** use a client certificate.

The `Frontend Trust Store` must contain the issuing CA.

NevisAdaptPluginPattern_nevisAdapt

Pattern reference for the nevisAdapt Instance to connect to.

SharedStorageSettings_storageSize

The size of the persistent volume. The minimum size is 1 gigabyte, we recommend to use at least 4 gigabytes.

For example: `4GB`

GroovyScriptStep_responseType

Choose between:

- `AUTH_ERROR` : terminates the session.
- `AUTH_CONTINUE` : use to produce a response and continue with this state on next request.

SamlSpConnector_onForbidden

Configure a step that shall be executed when the `Required Roles` check fails.

NevisFIDOConnector_url

Enter URL(s) to connect to your nevisFIDO instance.

The path must be omitted.

Only scheme `https://` is allowed.

The scheme is optional which means that you can enter simple `host:port` pairs (1 per line).

CSRFProtectionSettings_allowedDomains

CSRF protection can be obstructive for cross-domain use cases (e.g. federation or providing a public REST API).

Enter domains which should be excluded from `header-based` CSRF protection. There is no support for wildcards, pre- or postfix notations (sub-domains must be listed individually).

Example:

```
www.adnovum.ch
adnovum.ch
```

UserInput_variable

Enter `<scope>:<name>` of the variable which shall be set.

The following scopes are supported:

- `inargs`
- `notes`
- `sess` or `session`

For instance, enter `notes:loginid` to prefill the login form which is produced by the `nevisIDM Password Login` pattern.

SAPLogonTicket_setCookie

If set, this property must specify the value of the HTTP header "Set-Cookie". The cookie will be issued to the client by nevisAuth such that a cookie-based SSO federation with SAP applications is possible. This property is evaluated after the ticket has been issued, so the variables `sap.ticket` , `sap.ticket.maxAge` and `sap.ticket.expires` can be used.

Example value for this property that sets the cookie as expected by SAP products:

```
MYSAPSSO2=${outarg:sap.ticket}; Version=1; Path=/; Secure; HttpOnly; Max-Age=${notes:sap.ticket.maxAge}; Expires=${notes:sap.t
```

To use this example value by default, set this property to `true` .

NevisDetectAdminDeployable_jms

Add references (at least one) for the patterns configuring Java Messaging Service. In case of Kubernetes deployment, only one configuration is allowed.

Two different options are allowed at this time:

- `nevisDetect Message Queue Instance` - deployment pattern for a dedicated MQ component
- `ActiveMQ Client Configuration` - connect to an external ActiveMQ service via SSL

WARNING: In case of Kubernetes deployment, only `ActiveMQ Client Configuration` is supported.

DatabaseBase_hosts

Enter the host name of the database service.

The database service must be up when you deploy.

In a classic deployment the `Database User` and `Database Password` is used to connect.

In Kubernetes deployment a connection user and password will be generated and the `Root Credential` will be used to set up the database schema.

SecToken_attributes

Set the content of the `NEVIS SecToken`.

Example:

Attribute	Variable
userid	request:UserId
loginId	session:ch.nevis.session.loginid
profileId	session:ch.adnovum.nevisidm.profileId
clientId	session:ch.adnovum.nevisidm.clientId
roles	request:ActualRoles

Supported variable *scopes* are:

- `session` - a session variable.
- `request` - a variable from the request.
- `const` - a fixed value.

This configuration should work for most backend applications, including NEVIS components.

The `userid` is required by Ninja and **must** always be set. You can use `ch.nevis.session.loginid` when this pattern is **not** part of the `Initial Authentication Flow`.

The `loginId`, `profileId`, `clientId` are required by nevisIDM.

The attribute `roles` is required by nevisWF and nevisMeta.

For some attributes there are multiple variables to choose from. Check the nevisAuth log with log levels of `Vars` set to `INFO` to find out which variables are available in your case.

NevisFIDODeployable_managementPort

This port is used to check if the instance is up after deployment.

OAuth2Client_scopes

Enter scopes which may be requested by this client. The scope `openid` must be allowed when OpenID Connect is used.

FrontendKerberosLogin_keyTabFile

Upload the Kerberos keytab file.

nevisAuth uses this file to validate Kerberos tokens sent by browsers.

Please check the nevisAuth Technical Documentation on how to create the keytab file.

The keytab file will be deployed to the `conf` directory of the nevisAuth instance.

For a more secure and environment-specific configuration you have the following alternatives:

- create a variable and upload the keytab file in the inventory
- set `Keytab File Path` instead of uploading the file and deploy the file by other means

In complex setups with multiple `Kerberos Realms` and/or `Frontend Addresses` you may have to upload multiple keytab files.

DummyTAN_label

Set to show a different message.

InBandMobileAuthenticationRealm_keyStore

Define the key store to use for 2-way HTTPs connections from nevisProxy to nevisAuth.

HostContext_addons

Assign add-on patterns to customize the behaviour of this virtual host.

NevisFIDODatabase_maxConnectionLifetime

Defines the maximum time that a session database connection remains in the connection pool.

GenericIngressSettings_tlsSecrets

Use your own Kubernetes secrets to provide the frontend key store for a `Virtual Host`.

Syntax is a map of (primary) frontend address of the host to secret name.

```
www.siven.ch: customsecretname
```

Secrets must be of `type: kubernetes.io/tls`. Secrets must be prepared before deployment. They must contain a private key (`tls.key`), a matching certificate (`tls.crt`) and should contain the CA chain (`ca.crt`).

If not set the Nevis operator request certificates from the cluster issuer and generates a secret for each `Virtual Host` to store the required key material.

NevisIDMDatabase_oracleOwnerRoleName

Name of the owner role for the oracle database. It's recommended to keep the default value unless the pattern is used with an existing database that has a different one.

NevisFIDODeployable_deepLink

Deep links use the standard `https://` scheme.

Enter a complete URL here.

We recommend to add a **path** component as otherwise `/` will be used and there often is no appropriate content on the root location.

Note that **Apple requires the link to point to another domain**. You can use any web server or Nevis as the target.

When the user clicks the link, the OS of the mobile device will first try to download an *app link* file from a `/.well-known/` path on that domain.

You can configure `Deep Link Host` and `Deep Link App Files` to host these files.

The app link file is used by the OS to determine if a mobile app shall be opened, handing over the current URL.

If no mobile app can be determined, the deep link will be opened in the browser instead. Examples:

- user does not have the app installed
- no rule in the `/.well-known/apple-app-site-association` file applies to the path

Because of these error cases, there should be content on the deep link URL.

We recommend to create a page that informs the user how to install the mobile app. You can use the `Hosting Service` pattern to host this page on the `Deep Link Host` .

NevisIDMUserLookup_useDefaultProfile

Should in the Authentication flow assume default profile is selected if the user has multiple profiles, or should it display a selection dialog for the user.

DeployableBase_enforceVersion

Select `enabled` to perform basic version checks.

In classic VM deployment we run a command on each target host, to check which version of the component is installed.

In Kubernetes deployment we check the version of the docker image instead.

This check can be disabled for testing purposes.

GenericIngressSettings_clientCertVerifyDepth

The maximum validation depth between the provided client certificate and the CA chain. (default: 1).

You only need to increase this if you only have a parent CA in the CA Secret but want to accept client certificates which have been issued by a child CA.

WebApplicationAccess_path

Enter the path(s) where this application shall be accessible on the assigned `Virtual Host` .

It is recommended to set only 1 path. Examples:

- `/app/` - defines a base path. Requests which have a path component starting with `/app/` will be sent to this application. This is the most common scenario.
- `/` - may be used when there are no other applications. The `Hosted Resources` of the `Virtual Host` are still accessible but all other requests will be sent to the backend application.
- `exact:/app.html` - matches requests to `/app.html` only (query parameters may also be added). Use for single-page applications which don't require any additional resources.
- `prefix:/app` - matches requests which have a path component starting with `/app` . Examples: `/application` , `/app/index.html` , `/app2/secure/`

In case the frontend path is different from the path used within `Backend Addresses` then the path will be rewritten in incoming requests.

Note that for response by default only the headers are rewritten. See `Response Rewriting` for further options.

Note that when you enter multiple paths there are some limitations:

- Filters created by a `Realm` or `Additional Settings` will be mapped to all paths.
- The paths have to be the same on the backend server.

EmailTAN_channel

The connection provider for the sending the email code.

Choose between `Sendgrid SMTP` and a `Generic SMTP` patterns.

NevisIDMURLTicketConsume_onDisabled

Assign an authentication step to execute when the URL ticket or user is **disabled**.

If not set a screen with `title.url_ticket` and `error.user_or_url_ticket.disabled` will be shown in that case.

ServiceBase_addons

Assign add-on patterns to customize the behaviour of this service.

Example use cases:

- `Authorization Policy` to enforce roles or an authentication level.
- `URL Handling` to redirect or forward requests.
- `HTTP Header Customization` to add, replace, or remove HTTP headers in requests or responses.

AppleLogin_privateKeyFile

Private key provided by Apple. Find out more here.

If you upload your private key here and set the `Issuer` , the pattern will automatically generate the `Client Secret` .

If you do not want to configure your private key, you have to set the `Client Secret` instead.

AccessRestriction_defaultAction

Defines the action taken either when no country rules were matched or the IP of a request does not have an associated country in the database.

Possible actions are:

- **allow**: Requests are let through
- **log**: A log entry is made for each request
- **block**: Blocks requests

NevisIDMConnector_url

Enter URL(s) to connect to your nevisIDM instance.

The path must be omitted.

Only scheme `https://` is allowed.

The scheme is optional which means that you can enter simple `host:port` pairs (1 per line).

NevisIDMDeployable_resources

Files uploaded here will be added to the `conf` folder of the nevisIDM Instance.

NevisAuthRealmBase_sessionTimeout

Define the idle timeout of an authenticated session.

NevisIDMPasswordLogin_clientInput

Enable this to allow the user to enter the name of the *Client* (tenant) when logging in to nevisIDM.

If `disabled` , the input field is not shown and the Client `Default` is used.

OutOfBandMobileAuthentication_channel

Select how to transfer information to the mobile application.

`Link / QR-Code` : User can click a link if they are navigating on the same mobile device as the app resides on. A QR-code is shown as well which can be scanned if the user uses a browser separate from the mobile device the app resides on. The QR-code can also be scanned with the camera app of the mobile device. This option uses `mauth_link_qr.js` .

`Push / QR-Code (in-app)` : The user receives a push notification on their mobile device. In addition to that, a QR-code is shown which can be scanned instead in case the user does not receive the push notification. This QR-code can be scanned in the mobile app **only**, scanning the QR-Code using the camera app of the mobile device is not supported. This option uses `mauth_push_qr.js` .

If you are using a custom login template, add the correct Javascript file and some Velocity template snippets.

Download the default template in your `Authentication Realm` , unpack the zip, and search for `mauth` to get started.

NevisIDMPasswordLogin_redirectionValidationFallback

If `Allowed` then after checking regexes set in `Custom Redirection Path Validation Regexes` it also a check if the path starts with any declared `Web Application` 's path. (To see which paths would be find you can check Application Reports). If yes, those requests are also allowed

CustomNevisIDMLogFile_batchLogFormat

Log4j 2 log format for the default SERVER logs.

Note: not relevant when Log Targets is set to `syslog` .

NevisAdaptAuthenticationConnectorStep_passthroughMode

The passthrough mode disables the nevisAdapt validation. All analysers are still executed and results (risks/active sessions) are persisted.

When enabled, all risks follow the `On Success` step. High and Medium risk actions are ignored.

This mode is useful for data gathering and troubleshooting.

PropertiesTestPattern_hostPortProperty

Enter `host:port` pair(s).

BackendServiceAccessBase_host

Assign `Virtual Host` patterns which shall serve as entry point for this application.

NevisIDMPasswordLogin_passwordConfirmation

Select the behaviour of password reset and the form of the password reset screen. If

- `enabled` : displays password confirmation field on password reset screen which is required to be filled in for password to be reset.
- `disabled` : leaves out field on password reset screen and password can be reset with filling out password field only.

CustomProxyLogFile_rotationCompressionApplication

You may specify a program or script which shall be used to compress rotated files.

Example:

```
/usr/bin/gzip
```

AzureServiceBusRemoteQueue_host

Enter the complete `Host name` of the Service Bus as shown in the Azure portal.

Example: `my-service-bus-name.servicebus.windows.net`

GenericIngressSettings_clientCertErrorPage

An error page which will be presented in case of certificate validation error.

If you enter a path (e.g. `/errorpages/403.html`) then that path will be fetched from nevisProxy.

If you enter a URL then the caller is redirected to that URL.

NevisAuthRealmBase_keyStore

Define the key store to use for 2-way HTTPs connections from nevisProxy to nevisAuth.

If no pattern is assigned automatic key management will provide the required key material. This requires that the `nevisAuth Instance` is part of this project and also uses automatic key management.

Automatic key management should be used for test setups only.

GoogleLogin_clientExtId

The ExtId of the client in nevisIDM that will be used to store the user

NevisAuthRealmBase_trustStore

Defines the trust store that nevisProxy uses to validate the nevisAuth HTTPs endpoint.

If no pattern is assigned automatic key management is used to provide the trust store. This requires that the `nevisAuth Instance` is part of this project and also uses automatic key management.

Automatic key management should be used for test setups only.

NevisAuthRadiusResponse_attributes

Adds additional attributes to this Radius response.

Which attributes are required depends on the `Radius Response Type`.

For instance, in an `Access-Challenge` it is often required to add a `Reply-Message` which can be shown to the user. Also a `State` must be added some that the authentication can continue.

You may use expressions for the attribute value. For instance, use a `${litdict:` expression to return a translated text.

Examples:

```
Prompt: No-Echo
Reply-Message: ${litdict:mtan.prompt}
State: ${sess:Id}
```

OutOfBandMobileStepBase_onSuccess

On a successful authentication, the flow will continue with the assigned step.

GenericDeployment_commandTriggerFiles

Files deployed by other nevisAdmin 4 patterns that, when changed, trigger the script to be executed, even if the script and files itself do not change.

Example:

- /var/opt/nevisproxy/my_proxy/conf/navajo.xml

Hint: This is useful for patching e.g. `navajo.xml` after generation. Note that during the next deployment, it will be reverted (if the `nevisProxy Instance` pattern is deployed as well) and then patching will happen again.

NevisIDMDeployable_database

Assign a `nevisIDM Database` .

SwissPhoneChannel_username

The username to use to connect to the SwissPhone SMS Gateway.

NevisIDMDeployable_queryService

Enable the Query Service to allow full-text searches on the Admin GUI and REST API. Please note that using the Query Service requires the nevisIDM REST API to be exposed with the *nevisIDM REST Service* pattern.

NevisFIDOLogSettings_maxBackupIndex

Maximum number of backup files to keep in addition to the current log file. When `Rotation Type` is `time`, this property is used as Logback's maxHistory property. This means that logs will be archived for this number of time units where time unit is as defined in `Rotation Interval`.

AuthCloudBase_onSuccess

Assign a step to execute after successful authentication.

If no step is configured here the process ends and the user will be authenticated.

NevisProxyDeployable_startupDelay

Time to wait before checking Kubernetes readiness on startup.

You may have to increase this value if start of the nevisProxy service fails because of a failing readiness check.

Sets `initialDelaySeconds` of the Kubernetes startup probe.

AuthorizationPolicy_requiredRoles

Optional setting to enforce authorization.

Callers need any of the specified roles to access.

Required roles defined for an application can be overridden for a sub-path by combining several `Authorization Policy` patterns for this application. Required roles can also be inherited between patterns. See `Required Roles Mode` for details.

This setting requires assigning an `Authentication Realm` on the application pattern.

Usage examples:

- Enforce required roles for an application: use an `Authorization Policy` pattern with the `Required Roles` to enforce and link it to the application via `Additional Settings` ;
- Enforce required roles for some sub-paths of an application: use an `Authorization Policy` pattern with the `Required Roles` to enforce and `Apply only to sub-paths` set to the paths to protect. Link the pattern to the application via `Additional Settings` ;
- Enforce some main required roles for an application and some specific required roles for some sub-paths: use two `Authorization Policy` patterns, one with the main `Required Roles` and no sub-path, and one with the specific `Required Roles` and `Apply only to sub-paths` set to the paths where the specific required roles should apply. Link both patterns to the application via `Additional Settings` .
- Enforce some main required roles for an application and disable them for some sub-paths: use two `Authorization Policy` patterns, one with the main `Required Roles` and no sub-path, and one with no `Required Roles` and `Apply only to sub-paths` set to the paths where no required roles should be enforced. Link both patterns to the application via `Additional Settings` .
- Enforce some required roles for an application and add some forbidden roles for some sub-paths: use two `Authorization Policy` patterns, one with the `Required Roles` for the application, `Required Roles Mode` set to `self-contained` , and no sub-path, and the other pattern with no `Required Roles` , `Required Roles Mode` set to `inherited` , the `Forbidden Roles` for the subpaths, `Forbidden Roles Mode` set to `self-contained` , and `Apply only to sub-paths` set to the paths where the forbidden roles should be enforced. Link both patterns to the application via `Additional Settings` .

GenericDeployment_otherPermission

Read-write permissions for all users of the directory. All files and subdirectories (including unpacked from single .zip) will have the same permissions. The executable bit will be set automatically for readable directories and for readable `Executable Files` .

NevisIDMPasswordLogin_emailRedirRegexes

Enter regexes for Deny/Allow-list to validate redirection URL query parameter sent with the Password reset-email

Default defined for Deny-list regexes and filters out all paths containing `line feed` and `carriage return` characters.

GoogleLogin_clientSecret

Client Secret is `Client Secret` provided by Google when you create a OAUTH 2.0 credential in Google.

SocialLoginExtender_onSuccess

The step executed after a successful authentication. If no step is configured here the process ends with `AUTH_DONE` .

Note: In order to have profile selection in case account have multiple profiles, you need to use the User Lookup pattern.

GenericServiceSettings_filterMappings

Choose between:

- `manual` (default): only the `filter-mapping` elements which have been configured via `Filters and Mappings` will be added.
- `automatic` : filters configured via `Filters and Mappings` will be mapped to all `Frontend Paths` of the application.
- `both` : like `automatic` but additional `filter-mapping` elements are allowed as well.

CustomInputField_optional

Input into the field is optional or mandatory.

Choose between:

- `optional` - No input is required to the field.
- `mandatory` - Input is required to the field.

OAuth2Scope_scopeName

Enter the technical name of the scope.

If not set the name entered for the pattern will be used.

SamlIdp_host

Assign a `Virtual Host` which shall serve as entry point.

HostContext_unsecureConnection

This property defines how to handle requests received via plain HTTP. Choose between:

- `redirect` If a request is received via plain HTTP the client is redirect to the HTTPS endpoint (requires a `Frontend Address` with scheme `https://`).
- `allow` the request is processed.

NevisIDMDeployable_multiClientMode

If IDM should support multiple Clients.

BackendServiceAccessBase_loadBalancing

Select a request dispatching strategy when several `Backend Addresses` are configured.

- `disabled` - all requests will be sent to the first address. If this address is not available the next address is chosen;
- `round-robin` - one of the addresses will be picked up for each request using a round-robin rotation;
- `session-sticky` - one of the addresses will be picked up for each new session using a round-robin rotation, then subsequent requests for the session will be sent to the same address.

Failover strategy:

- When the selected backend cannot be accessed, nevisProxy will attempt to use another one.
- Once the said backend can be accessed again, it can be picked up for new requests if the load balancing is `round-robin` , or for new sessions if the load balancing is `disabled` or `session-sticky` . The requests linked to an existing session will still go to the current backend until the end of the session if the load balancing is `disabled` or `session-sticky` .

NevisAdaptAuthenticationConnectorStep_clientTrustStore

The trust store used by this pattern to establish a connection with the nevisAdapt component. This trust store must trust the `nevisAdapt Instance` 's key store. Please reference a trust store provider pattern or leave empty to manage the trust store with nevisAdmin automatic key management.

NevisIDMChangePassword_locked

Assign an authentication step to execute when the status of the URL ticket or credential is **locked**.

CustomAuthLogFile_eventLogFields

Set to add additional fields to the nevisAuth events log.

Enter field names (with optional format `JSON`) to nevisAuth expressions.

Examples:

Field	Expression
unitDisplayName	`${sess:ch.nevis.idm.User.unit.displayName}`
unitHierarchy:JSON	`${StringUtils.strip(sess['ch.nevis.idm.User.unit.hname'].replace('/',','),',')}`

Based on this `CustomField` elements with `name`, `value`, and optional attribute `format`, are created in `esauth4.xml`.

NevisFIDOServiceAccess_backendKeyStore

Assign a key store for 2-way TLS connection to nevisFIDO.

SwissPhoneChannel_proxy

Forward proxy for the connection to the SwissPhone SMS Gateway.

Example: `proxy.your-internal-domain:3128`

NevisIDMTermsAcceptance_onSuccess

Configure the step to execute after the user has accepted all terms and conditions.

If no step is configured here the process ends and the user will be authenticated.

KerberosLogin_keyTabFilePath

Enter the path of the Kerberos keytab file.

The path must exist on the target host(s) of the `nevisAuth Instance`.

This configuration is ignored when keytab file(s) are uploaded via `Keytab File`.

In complex setups with multiple `Kerberos Realms` and/or `Frontend Addresses` you may want to enter multiple keytab file paths.

NevisIDMServiceAccessBase_backendHostnameCheck

Enable to verify that the hostname on the certificate presented by the backend matches the hostname of `nevisIDM`

NevisAuthRealmBase_addons

Assign `Session Settings` to set advanced settings, such as session timeout and session validation requirements.

GenericAuthRealm_labels

Labels are used to provide human-readable text in the language of the user.

The language is extracted from the `Accept-Language` header and the default login page template has a language selection.

Which labels are used depends on the assigned steps. Click `Download Default Labels` to retrieve the labels used and their translations.

Here you can overwrite the defaults and add your own translations or even introduce new labels which may be required when using a `Custom Login Template` or `Generic Authentication Step` patterns.

The name of uploaded files must end with the language code. As the format is compatible you may upload existing `text_<code>.properties` files of nevisLogrend or `LitDict_<code>.properties` of nevisAuth.

The encoding of uploaded files does not matter as long as all translations are HTML encoded.

The default login template uses the following labels:

- `title` - used as browser page title
- `language.<code>` - used by language switch component

The default logout process of nevisAuth (which will be applied when no step is assigned to `Logout`) produces a confirmation GUI which requires the following labels:

- `logout.label` - header of the logout confirmation GUI
- `logout.text` - text shown to the user
- `continue.button.label` - label on the confirmation button

NevisDetectAdminWebApplicationAccess_backendHostnameCheck

Enable to verify that the hostname on the certificate presented by the backend matches the hostname of `nevisDetect Admin`

NevisAdaptRememberMeConnectorStep_clientKeyStore

The key store used by this pattern to establish a connection with the nevisAdapt component. For a client TLS connection, this key store should be trusted by the `nevisAdapt Instance` . If no pattern is assigned here automatic key management will provide the key store.

PropertiesTestPattern_simpleTextProperty

Enter text. Each line is one value.

HostContext_requireClientCert

Choose from:

- `disabled (default)` : No client certificate is required to connect to this virtual host.
- `enabled` : Clients must present a client certificate signed by a CA. The CA which has issued the client certificate must be part of the `Frontend Truststore` . When no client certificate is presented or the certificate is not valid the connection will be aborted. As no error page is rendered this feature is not recommended when there are browser-based clients. Use for technical clients only.

NevisAdaptDeployable_proxyPort

Enter the port of the forward proxy if available.

```
3182
```

AccessTokenConsumer_onSuccess

Assign a step to continue with after successfully validating the token.

NevisIDMUserLookup_mode

Select `interactive` to prompt the user to enter a Login ID.

An input form will be shown when the query or POST parameter `isiwebuserid` is missing or the user is not found in nevisIDM (and `On User Not Found` is not set).

Select `pass-through` to look up the user based on `Login ID Source`.

In this mode no input form will be shown. Instead, a `403` response will be generated if the user is not found (and `On User Not Found` is not set).

GenericAuthService_path

Define a path to be mapped on the assigned virtual host.

Requests sent to this path will be forwarded to nevisAuth so that they can be handled by this authentication service.

SamlSpConnector_authenticationLevel

Enforce a minimum required authentication level for this Service Provider.

If not set, the minimum required authentication level will depend on the incoming `AuthnRequest`. An SP may specify the level by including a `RequestedAuthnContext`, such as:

```
<samlp:RequestedAuthnContext Comparison="minimum">
  <saml:AuthnContextClassRef>urn:nevis:level:2</saml:AuthnContextClassRef>
</samlp:RequestedAuthnContext>
```

If there is any requirement for a minimum authentication level, the `Authentication Realm` must provide a `Session Upgrade Flow` for that level. See the help of the `Authentication Realm` pattern for details.

Note that when there is no authenticated session, the `Initial Authentication Flow` of the `Authentication Realm` will be executed first.

After successfully completing the `Initial Authentication Flow` the attained authentication level is compared against the minimum level and, if required, a `Session Upgrade Flow` is executed.

AzureServiceBusRemoteQueue_queue

Enter the name of a queue.

OAuth2AuthorizationServer_signer

Configure the key material which is used to sign issued codes and tokens.

NevisAdaptFeedbackConfig_feedbackAction

The authentication step is able to generate a short-term feedback token if there are suspicious circumstances around the authentication attempt.

The registered user receives a URL in a notification email (in a notification step if configured), following that link within the token's lifetime would perform the configured task:

- `disabled` - no token will be generated
- `session` - following the link distrusts the suspicious session (even retroactively)
- `device` - following the link distrusts the suspicious session and all other sessions associated with the same device
- `all` - following the link removes all sessions and observations for the user

All options apart from `disabled` require access to SessionManagement API in all involved `nevisAuth Instance`.

In case of `all`, please set `Enable Indexing` value to `on` for all involved `nevisAuth Instance`.

AuthCloudBase_onFailure

Assign a step to continue with when the operation has failed due to unknown reasons.

For instance, you may assign the following steps:

- `User Information`: show an error message and terminate the authentication flow.
- `nevisIDM Second Factor Selection`: select an alternative second factor for authentication.

CustomNevisMetaLogFile_serverLog

Select type of log4j appender.

`RollingFileAppender` and `SyslogAppender` are possible options.

NevisProxyObservabilitySettings_traceMode

Choose one of:

- **enabled**: enable the trace feature of OpenTelemetry
- **disabled**: disable the trace feature of OpenTelemetry

NevisMetaDeployable_managementPort

This port is used to check if the instance is up after deployment.

OutOfBandMobileAuthentication_onDispatchFailure

When a failure occurs during *dispatching*, the authentication flow will continue with the assigned step.

There are several error cases:

- nevisFIDO is unable to hand out a link or render a QR-code
- the `dispatchTargetId` sent by the JavaScript does not exist. For instance, the credential may have been deleted in nevisIDM.

NevisDPDeployable_idmTruststore

Assign a trust store which shall be used for outbound TLS connections to nevisIDM. If no pattern is assigned no trust store will be generated.

For nevisDataPorter to use the trust store, the following expressions should be used inside the `dataporter.xml` file:

```
${idm.truststore}
${idm.truststore.password}
```

Example configuration:

```
<object type="NevisIDMConnectionPool" name="adminService">
    <dp:paraVal name="endpoint" value="${cfg.idmEndpoint}"/>
    <dp:paraVal name="loginMode" value="proxyCert"/>
    <dp:paraMap name="sslSettings">
        <value name="javax.net.ssl.trustStore" value="${idm.truststore}"/>
        <value name="javax.net.ssl.trustStorePassword" value="${idm.truststore.password}"/>
        ...
    </dp:paraMap>
</object>
```

NevisAuthDeployable_database

By default, nevisAuth stores sessions and out of context data in memory.

In most setups you should use a database instead, and you should assign a `nevisAuth Database` pattern here.

In memory should be used only when there is only 1 line / pod of nevisAuth, or in a classic deployment where nevisProxy can ensure session-sticky load balancing towards nevisAuth.

OAuth2AuthorizationServer_setupId

ID of the nevisMeta *setup*.

Create your setup via the `nevisMeta Web Console`.

Then the ID of the setup can be determined. There are several ways to do that:

- hover over the icon which links to the REST API
- export the setup and check the exported files
- Configure a `nevisMeta REST Service`, login and send a `GET` to `/nevismeta/rest/v2/modules/oauthv2/setups/`

NevisIDMDeployable_managementHost

Enter a custom host name to open the `Status Port` on.

If not set `0.0.0.0` will be used in case of Kubernetes deployment and `localhost` for deployment to VMs.

CustomNevisMetaLogFile_maxBackupIndex

Maximum number of backup files to keep in addition to the current log file.

This setting applies to `nevismeta.log` only.

OAuth2AuthorizationServer_oidcIssuer

Enter the *issuer* for OpenID Connect.

The value must be a case-sensitive URL using the https scheme that contains at least scheme and host. The port number and path component are optional. No query or fragment components are allowed.

If not set the issuer will be calculated based on:

- the first `Frontend Address` with scheme `https` of the assigned `Virtual Host`
- the first `Frontend Path`

NevisIDMPasswordLogin_nevisIDM

Reference a `nevisIDM Instance` to be used for the username / password authentication.

WebSocket_params

Add custom `init-param` for the WebSocket servlet.

Please check the nevisProxy technical documentation for supported `init-params` of the servlet class `ch::nevis::isiweb4::servlet::connector::websocket::WebSocketServlet`.

SamlSpConnector_subject

Set to use a different subject for the SAML `Assertion`.

Examples:

- `${sess:ch.nevis.session.loginid}` - what the user has entered to login

TokenHeaderPropagation_token

Assign a Token pattern.

The referred pattern must be assigned to the correct Realm pattern(s).

AuthServiceBase_allowedMethods

Define the allowed HTTP methods.

If not configured, all HTTP methods are allowed.

8.2405.0

Full changelog:

Patterns 8.2405.0 Release Notes - 2024-05-15

Removed FIDO Facets

The following 2 values have been removed from the default facets in nevisFIDO UAF Instance:

- `android:apk-key-hash:z7Xkw62dAn/BsckOQ9a3OMhmlwhzdr2VkcswIIyJgJE`
- `ios:bundle-id:ch.nevis.accessapp.presales.k8s`

If your app uses any of these facets you have to configure them.

TransformVariablesStep_onSuccess

Set the step to continue with after successful execution.

HostContext_crsVersion

Allows to select the OWASP ModSecurity CRS version.

Available options are:

- `4.7.0` : newest version of CRS, uses Anomaly Scoring Mode, minimal CRS setup
- `3.3.5` : default and recommended setup, uses Anomaly Scoring Mode
- `3.3.2` : previous version of CRS, uses Anomaly Scoring Mode, kept for easier migration
- `custom` : allows to upload a custom rule set. See the `ModSecurity Rule Set` option for more information.

The following HTTP methods are allowed by default when selecting any of the included versions:

```
CHECKOUT, COPY, DELETE, GET, HEAD,
LOCK, MERGE, MKACTIVITY, MKCOL, MOVE, OPTIONS,
POST, PROPFIND, PROPPATCH, PATCH, POST, PUT, TRACE, UNLOCK
```

SamlIdpConnector_decryptionKey

Assign a pattern to configure the private key to decrypt the incoming message of the identity provider.

RealmBase_cookieSameSite

In February 2020 Chrome 80 has been released which treats cookies without SameSite flag as `Lax` .

This change can break cross-domain use cases (e.g. SAML).

Thus, it is recommended to select `None` here.

If `None` is selected, and you have to support older browsers also check `Cookie Same Site Relaxation` .

If you do not expect any requests from other domains, you may also go for `Lax` or `Strict` as this increases security.

NevisIDMDeployable_mailSenderAddress

The default sender address for e-mails.

NevisAuthRealm_auth

Assign a `nevisAuth Instance` pattern.

ApplicationProtectionDeployableBase_bindHost

Enter a custom host name to listen on.

This setting is relevant in classic VM deployment, when working with multi-homed target hosts.

In Kubernetes the component listens on `0.0.0.0` and thus this setting is discouraged.

URLHandler_redirects

Terminate requests by returning a *HTTP Redirect* (status code `302`).

In the first column (*source*) enter the current location. In the second column enter the destination to redirect to.

The following formats are supported:

- `URL`
- `absolute path` (starting with `/`)
- `relative path`

Regular expressions are supported in the source, and group extractions may be used in the destination.

Absolute paths always point to the host, while relative paths are appended to the path of the assigned host (`/`) or application.

The order of the rules matters. Only the first matching rule is applied.

Examples:

Source	Destination	Description
`http://(.*)`	`https://$1`	redirects plain HTTP to HTTPs, preserving the request path
`(.*)?lang=de`	`de/$1`	put query parameter into request path
`/nevis.html`	`https://www.nevis.ch`	redirect requests to a certain HTML page to a different domain

NevisMetaDatabase_type

Choose between `MariaDB` and `PostgresSQL`.

We recommend to use `MariaDB` as it is supported by all Nevis components that have a database.

Note: `PostgresSQL` database is only experimental configuration.

NevisAuthDatabase_type

Choose between `MariaDB` and `PostgresSQL`.

We recommend to use `MariaDB` as it is supported by all Nevis components that have a database.

Note: `PostgresSQL` database is only experimental configuration.

SamlIdpConnector_transitions

Add or overwrite `ResultCond` elements in the `ServiceProviderState` state.

This setting is advanced. Use without proper know-how may lead to incorrect behavior.

If you use this setting, we recommend that you contact Nevis to discuss your use case.

The position refers to the list of `Additional Follow-up Steps`. The position starts at 1.

Examples:

`ResultCond`	Position
status-Responder	1
status-Responder-AuthnFailed	2

The following `ResultCond` elements cannot be overruled by this setting:

- `ok`
- `logout`
- `logoutCompleted`
- `logoutFailed`

NevisKeyboxProvider_validation

Allows to the validation in case the nevisKeybox is deployed by this project (e.g. using `Generic Deployment`).

SocialLoginExtender_onFailure

The step that will be executed if the authentication fails. If no step is configured here the process ends with `AUTH_ERROR` .

NevisProxyDeployable_cacheSize

Configures the approximate size of the Apache SSL Cache.

The minimum allowed value is `1 KB` . The maximum is `100 MB` .

If not, the default from Apache is used.

NevisAdaptAuthenticationConnectorStep_onMediumRisk

Will be considered only if `Profile` is set to either `balanced` , `strict` or `custom` .

Set the step to continue with if the calculated risk score exceeds the Medium threshold.

In case it remains unset:

1. `On High Risk` becomes mandatory
2. Applies the same next step as `On Success`

LdapLogin_onInvalidPassword

Assign an authentication step to be processed if the user is found but the password is incorrect.

Use for custom reporting or error handling.

If no step is assigned the GUI is displayed again and an error message will be shown.

This setting is experimental and may be adapted in future releases.

GenericAuthXmlServiceBase_auth

Assign a `nevisAuth Instance` .

WebhookCalls_onFailure

Assign the next authentication step (optional).

SamlToken_attributes

Define which nevisAuth `session variables` to include as `attributes` in the SAML assertion.

If not set the following default will be used:

Attribute	Session Variable
userid	ch.nevis.session.userid
loginId	ch.nevis.session.loginid
profileId	ch.adnovum.nevisidm.profileId
clientId	ch.adnovum.nevisidm.clientId

Which session variables are available depends on your authentication flow. For instance, if you use `nevisIDM Password Login` there will be a session variable `user.email` so you can easily add an attribute `email` .

Set the log level of `Vars` to `INFO` and check the `esauth4sv.log` to find out which session variables are available after authentication.

In case a session variable is not found the attribute will be omitted.

NevisProxyDeployable_defaultHostContext

The default virtual host of this nevisProxy instance.

The default will be used for requests without a `Host` header or if there is no host with a corresponding frontend address.

MultipleFieldUserInput_buttons

Assign an `Dispatcher Button` to add a button which points to a different authentication step.

NevisAdaptRestServiceAccess_adapt

Reference to the nevisAdapt Instance pattern.

PemTrustStoreProvider_dirName

Enter a name for the trust store directory which is used instead of the pattern name.

This configuration may be used to prevent trust stores overwriting each other and is only required in complex setups with multiple projects or inventories.

ServiceBase_token

Propagate a token to the backend application. The token informs the application about the authenticated user.

For instance, assign `NEVIS SecToken` if the application uses Ninja or `SAML Token` for applications which are able to consume SAML Responses.

NevisIDMDatabase_parameters

Enter parameters for the DB connection string.

Enter 1 parameter per line.

Lines will be joined with `&`.

When connecting to a MariaDB database some query parameters will be added when not present. The following parameters will be enforced then:

```
pinGlobalTxToPhysicalConnection=1
useMysqlMetadata=true
cachePrepStmts=true
prepStmtCacheSize=1000
```

KeyObject_revocation

Define the `revocation` attribute of the `KeyObject`.

You can enter a path or URL of the certificate revocation list or the URL to the OCSP service.

See Generic key material configuration attributes for examples.

ApplicationProtectionDeployableBase_secTokenTrustStore

Assign the Trust Store provider for verifying the NEVIS SecToken. If no pattern is assigned the signer key will be provided by the nevisAdmin 4 PKI.

GenericAuthenticationStep_resources

Upload additional configuration files or scripts required by your `AuthState` configuration. Uploaded files will be deployed into the `conf` directory of the nevisAuth instance.

CustomProxyLogFile_conditionalLogLevels

Can be used to configure log levels based on conditions.

Example:

```
Condition:REMOTE_ADDR:CIDR/10.4.12.0/24/
Pragma: block-begin
BC.Tracer.DebugProfile.NavajoOp=4
BC.Tracer.DebugProfile.IsiwebOp=4
BC.Tracer.DebugProfile.IW4IdentCreaFlt=4
Pragma: block-end
```

NevisProxyDeployable_restartPolicy

Determines the instance behaviour when a configuration change triggers an optional restart.

Select one of:

- `eager` - the instance will restart when deploying the new configuration;
- `lazy` - the instance will skip optional restarts.

OAuth2AuthorizationServer_authorizationEndpoint

This is the path where relying parties redirect the browser to.

Example use cases:

- **OAuth**: acquire an access and refresh tokens
- **OpenID Connect**: acquire access, refresh and ID tokens

Use the `exact:` prefix to expose only the given path. Without this prefix sub-paths will be accessible as well. This is because a normal mapping with `/*` at the end will be created in nevisProxy.

HeaderCustomization_requestHeadersRemove

Removes HTTP headers from requests.

The syntax is: `<header name>`

Examples:

```
User-Agent
```

Note: change the `Filter Phase` to remove headers early / late.

SocialLoginBase_onUserNotFound

Configure the authentication flow to be executed when no user was found and the email provided by social account does not exist. The authentication flow must contain the `Social Login Create User` pattern if a new user shall be created.

Note: Please select scope `email` and `profile` for getting user's information from social account.

FIDO2Onboarding_attestation

Define the preference for attestation conveyance.

You can configure if you want an attestation statement.

- `none` - no attestation statement required.
- `direct` - receive an attestation statement as produced by the authenticator.
- `indirect` - requests an attestation statement but allows the client to modify what has been received from the authenticator (e.g. for anonymization).

GenericDeployment_parameters

Define *Template Parameters*.

Examples:

```
smtp: smtp.siven.ch
sender: noreply@siven.ch
```

These parameters can be used in:

- uploaded files matching an expression specified in the `Template Files` property
- the value of the `Path` property
- the value of the `Command` property

The expression formats are:

`${param.<name>}`:

- `name` found: parameter value is used.
- `name` missing: expression is **not** replaced.

`${param.<name>:<default value>}`:

- `name` found: parameter value is used.
- `name` missing: default value will be used.

In `<default value>` the character `}` must be escaped as `\}`.

UnauthenticatedRealm_timestampInterval

Sets the minimum time interval between two updates of the session timestamp.

If the parameter is set to "0", the system will update the session timestamp each time a request accesses a session.

The `Initial Session Timeout` is used as `Update Session Timestamp Interval` if it is shorter than the duration configured here.

NevisFIDODeployable_idm

Assign a `nevisIDM Instance` or `nevisIDM Connector` pattern.

Use `nevisIDM Connector` only when the nevisIDM instance is not setup by the same nevisAdmin 4 project.

When using `nevisIDM Connector` you have to use non-automatic key management.

DeployableBase_deploymentHosts

The host group or Kubernetes service that this instance will be deployed to. For testing purposes you can also enter a host name instead of a group.

For classic deployment, the host name / group must exist in the selected inventory. For Kubernetes deployment, defining the service is optional in the inventory.

NevisAuthRealmBase_defaultLabels

Choose between:

- `enabled` - add default translations for labels which are commonly used (e.g. `title` or language labels in nevisLogrend, error labels in nevisAuth) and which are required by the realm patterns (e.g. assigned authentication steps).
- `disabled` - select to only add what has been uploaded via `Translations`. Note that if `Translations` are incomplete users may encounter untranslated labels.

NevisAdaptDeployableBase_clientAuth

Setting for 2-way TLS on the nevisAdapt HTTPs endpoint. There are 3 options will affect the callers (e.g. nevisProxy or technical clients accessing nevisAdapt REST APIs)

- required: Callers **must** present a client certificate.
- requested: Callers **can** present a client certificate.
- disabled: Callers **must not** use a client certificate.

The `Frontend Trust Store` must contain the issuing CA.

NevisDetectPersistencyDeployable_port

Enter the port on which nevisDetect Persistency will listen.

NevisIDMUserLookup_loginIdSource

Enter a *nevisAuth expression* for the login ID which is used to look up the user.

Supported and required in authentication mode `pass-through` only.

Examples

- `${inargs:isiwebuserid}`

NevisIDMGenericBatchJob_trigger

Add configuration of a bean which acts as a trigger for job execution.

Execute every 24 hours:

```
<bean id="someTriggerId" class="org.springframework.scheduling.quartz.SimpleTriggerFactoryBean">
    <property name="description" value="Some description shown in nevisIDM Admin GUI"/>
    <property name="jobDetail" ref="someJobId"/> <!-- must be provided via Job(s) -->
    <property name="repeatInterval" value="86400000"/> <!-- 1 day in ms -->
    <property name="misfireInstructionName" value="MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_EXISTING_COUNT"/>
</bean>
```

Execute once a day at midnight (cron expression):

```
<bean id="someTriggerId" class="org.springframework.scheduling.quartz.CronTriggerFactoryBean">
    <property name="description" value="Some description shown in nevisIDM Admin GUI"/>
    <property name="jobDetail" ref="someJobId"/> <!-- must be provided via Job(s) -->
    <property name="cronExpression" value="0 0 0 * * ?"/>
</bean>
```

InBandMobileDeviceRegistration_nevisfido

Assign a nevisFIDO instance.

This instance will be responsible for providing the device registration services.

NevisIDMConnectorAddon_nevisIDM

The nevisIDM instance that the generated `AuthState` should connect to.

NevisIDMURLTicketConsume_onNotFound

Assign an authentication step to execute when the URL ticket is **not found**.

If not set a screen with `title.url_ticket` and `error.url_ticket.not_found` will be shown in that case.

TANBase_guiName

Change the `name` of the `Gui` element.

Change this only if you need the Gui name your login template to render the screen differently.

MobileDeviceDeregistration_nevisfido

Assign a `nevisFIDO UAF Instance`. This instance will be responsible for providing the mobile device deregistration services.

NevisLogrendDeployable_keyStore

Used when simple or mutual (2-way) HTTPs is configured. If no pattern is assigned here automatic key management will provide the key store.

NevisAuthRealmBase_labelsMode

Choose between:

- `combined` - upload 1 file per language code named `labels_<code>.properties` . The labels will be added to both nevisAuth and nevisLogrend. Alternatively, you can upload a zip file called `labels.zip` containing these properties files.
- `separate` - select *only* when you need different labels in nevisAuth and nevisLogrend. The files must be called `LitDict_<code>.properties` for nevisAuth and `text_<code>.properties` for nevisLogrend. Alternatively, you may upload zip file called `LitDict.zip` and `text.zip` containing these properties files.

SamlIdp_sp

Define the SAML Service Providers which can use this IDP.

For each SP an own `AuthState` of class `IdentityProviderState` will be generated.

NevisProxyDeployable_restartCondition

Enter an expression to prevent nevisProxy from being restarted even if the configuration changes.

nevisProxy will only be restarted if the exit status is `0` .

The expression must always terminate.

In Kubernetes deployment this setting is ignored.

A use case where this is required is when nevisProxy is deployed to multiple hosts and listens on a shared IP which is bound on 1 host only.

```
ip address show dev eth1 | grep -q "172.29.0.5"
```

Example for multiple shared IPs:

```
ip address show dev eth1 | egrep -q "172.29.0.5|172.29.0.6"
```

Recommendations:

- Run the command manually on the target host to be sure that it works for you.
- You can check the exit status of the last command by running `echo $?`

NevisMetaServiceAccessBase_realm

Assign a realm pattern which authenticates access to nevisMeta.

KerberosLogin_onFailure

Assign authentication step that is processed if Kerberos authentication fails.

If no step is assigned an AuthState `Authentication_Failed` will be created automatically.

HostingService_defaultFile

Defines a default file which will be returned when there is no other matching file.

SwissPhoneChannel_backendTrustStore

Assign a trust store for the outbound TLS connection to SwissPhone.

Import the CA certificate of the `Portal Server` into this trust store.

Since version 4.38 nevisAuth trusts CA certificates included in the JDK.

Thus, it is not required to configure this.

However, you can still configure a trust store here to be as strict as possible.

AuthCloudBase_skipType

The type of element which allows the user to skip this step.

The element is usually a button but may also be changed to an `info` text. As info elements may contain HTML you can display a link that behaves like a button.

NevisDetectLogSettings_maxBackupIndex

Maximum number of backup files to keep in addition to the current log file. When `Rotation Type` is `time` , this property is used as Logback's maxHistory property. This means that logs will be archived for this number of time units where time unit is as defined in `Rotation Interval` .

OAuth2UserInfo_host

Assign a `Virtual Host` which shall serve as entry point.

SamlIdp_logoutConfirmation

Choose between:

- `enabled` - shows a logout confirmation screen when the path ends with `/logout`
- `disabled` - never shows a logout confirmation screen

Please be aware that we plan further changes which affect SAML logout and thus this setting may change or even disappear in a future release.

Dispatcher_conditions

Configure *conditions*.

The first column gives your condition a name. The name must be unique and must be used in `Transition(s)` .

In the second column enter an *expression*. This may be a nevisAuth expression (`${...}`) or EL expression (`#{...}`). See nevisAuth Technical Documentation for information about the expression syntax.

In EL expressions it is possible to reference variables from the inventory, an example can be found below.

All conditions will be evaluated and thus multiple conditions may apply. In this case the combination of conditions in must be configured in `Transition(s)` .

Examples:

Key	Value
pwreset	`${request:currentResource:/pwreset:true}`

Key	Value
sp	`${sess:ch.nevis.auth.saml.request.issuer:^SP$:true}`
mfa	`#{${var.mtanEnabled} or ${var.oathEnabled}}`

OAuth2Client_type

Reserved for ID Cloud use.

CustomNevisMetaLogFile_regexFilter

If set, messages for `nevismeta.log` which match the given regular expression won't be logged.

The regular expression must match the entire line. For instance, you may use the following format to match `some text` :

```
.*some text.*
```

NevisIDMPasswordCreate_credentialState

The state which the credential is in when created. Options:

- INITIAL
- ACTIVE
- DISABLED

NevisIDMWebApplicationAccess_requestValidation

- `off` - no request validation
- `standard` - uses ModSecurity OWASP Core Rule Set (CRS) with default paranoia level 1 - Basic security
- `custom` - configure `Request Validation Settings` via `Additional Settings`
- `log only` - uses `standard` in log only mode

NevisDetectEntrypointDeployable_port

Enter the port on which nevisDetect Feature Correlator will listen.

CustomProxyLogFile_logFormat

Allows the configuration of the `LogFormat` Apache directive in the navajo.xml file.

For more information, check the official Apache documentation of the directive.

NevisAuthDeployable_idPregenerate

Define the value of the `idPregenerate` attribute.

- `enabled` : `true` is set.
- `disabled` : `false` is set.

Do not change this unless you know what you are doing.

WebhookCalls_onSuccess

Assign the next authentication step (optional).

ServiceAccessBase_allowedMethods

Define the HTTP methods allowed for this application.

Methods which are listed here must also be allowed on the `Virtual Host`.

You may also use the following method groups:

- `ALL-HTTP` includes common HTTP methods.

  These are: `GET`, `POST`, `HEAD`, `DELETE`, `TRACE`, `CONNECT`, `OPTIONS`, `PUT`, `PATCH`

- `ALL-WEBDAV` includes all methods required for WebDAV.

  These are: `MERGE`, `UNCHECKOUT`, `MKACTIVITY`, `PROPPATCH`, `LOCK`, `CHECKOUT`, `SEARCH`, `COPY`, `MKCOL`, `MKWORKSPACE`, `PROPFIND`, `UPDATE`, `REBIND`, `BASELINE-CONTROL`, `UNBIND`, `CHECKIN`, `VERSION-CONTROL`, `UNLOCK`, `LABEL`, `MOVE`, `ACL`, `BIND`, `REPORT`

To remove methods from `ALL-HTTP` and `ALL-WEBDAV` simply add the method with a `-` sign in front of it.

NevisIDMDeployable_defaultLanguage

Sets default language for nevisIDM.

It is the same as using `web.gui.languages.default` in `properties`. If given by both way, the value in `properties` will be used.

See nevisIDM Reference Guide (chapter Configuration files) for details.

AuditChannel_properties

Provide configuration for the audit channel.

For each key-value pair 1 `property` element will be generated.

Button_onClick

Assign an authentication step to continue with when the button is clicked.

NevisIDMDeployable_mailSMTPHost

The name of the host on which the SMTP server is running.

TCPSettings_keepAliveLifetime

The absolute lifetime of a TCP connection. This should be configured to less than the connection lifetime allowed by the firewall between nevisProxy and the content providers. By leaving this field empty, you will be using the nevisProxy default value.

SamlToken_properties

Enter custom properties for the nevisAuth `IdentityProviderState` which issues the SAML `Response` (or `Assertion`).

Please check the technical documentation for details.

Common use cases are:

- `out.issuer` : sets the `Issuer` element (By default, the sanitized name of the pattern is used)
- `out.audienceRestriction` : some recipients require this to be set to decide if they accept the token
- `out.signatureKeyInfo` : add information about the signer certificate

Examples:

```
out.authnContextClassRef = urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
out.sessionIndex = ${notes:saml.assertionId}
out.signatureKeyInfo = Certificate
out.subject.format = urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified
out.ttl = 10
relayState = ${request:currentResource}
```

InBandMobileAuthenticationRealm_signerTrustStore

Defines the trust store nevisProxy uses for validating the signature of the NEVIS SecToken issued by nevisAuth.

NevisDetectDeployableBase_jmsClientKeyStore

Used when simple or mutual (2-way) HTTPs is configured. If no pattern is assigned here automatic key management will provide the key store.

NevisMetaDeployable_database

Assign a `nevisMeta Database` .

NevisIDMDatabase_oracleDataTablespaceName

Name of the data tablespace for the oracle database. It's recommended to keep the default value unless the pattern is used with an existing database that has a different one.

SamlSpConnector_sign

Configure what to sign.

The setting `none` is not recommended for productive setups as it is vulnerable to attacks.

The setting `Assertion` may require additional checks on service provider side to close the attack vector. For instance, count the number of `Assertion` elements in the message.

When `recommended` is selected the signed element depends on the `Outbound Binding`:

- `http-redirect`: nothing will be signed as signing is not supported.
- `http-post`: the `Response` is signed as this is most secure.

NevisAuthRadiusFacade_inputs

Define how attributes from Radius requests are mapped to input arguments (`inargs`) for nevisAuth.

There are some well-known input arguments which you may have to provide:

- `isiwebuserid` - entered user name
- `isiwebpasswd` - entered password
- `mtanresponse` - used in mobile TAN patterns

Examples:

Radius Attribute	Input
User-Name	`isiwebuserid`
User-Password	`isiwebpasswd` , `mtanresponse`
State	`sessionid`

Depending on your authentication flow it may be required to provide additional input arguments.

GenericWebBase_phase

When adding `filter-mapping` elements, a phase must be defined.

The phase defines where the `filter-mapping` is placed in the `web.xml` and ensures that filters are applied in the right order, relative to other phases.

The order within a certain phase is undefined as it must **not** matter.

The order for requests is `START` to `END` and `END` to `START` for responses.

This setting applies to all `filter-mapping` elements.

The `filter-mapping` elements may be provided via *Filters and Servlets*, or created automatically (see `Filter Mappings` for details).

Choose from the following filter phases:

- `START` : applied as early as possible for requests and as late as possible for responses.
- `BEFORE_SANITATION` : applied before filters which validate the request (e.g. Mod Security).
- `SANITATION` : used for security. This is the first phase which allows accessing the session for applications protected by a realm.
- `AFTER_SANITATION` : your request has passed security checks.

- `BEFORE_AUTHENTICATION` : applied just before authentication.
- `AUTHENTICATION` : used by the filter which connects to nevisAuth for applications which are protected by an `Authentication Realm` .
- `AFTER_AUTHENTICATION` : the request has level 1 authentication. Used by `Authorization Policy` for `Authentication Level` stepup.
- `BEFORE_AUTHORIZATION` : choose this phase to do preprocessing before authorization.
- `AUTHORIZATION` : used by `Authorization Policy` for `Required Roles` check.
- `AFTER_AUTHORIZATION` : used by patterns assigned as `Application Access Token` to applications.
- `END` : applied as late as possible for requests and as early as possible for responses.

This setting is ignored when you patch a `filter` generated by another pattern (e.g. by adding, overwriting, or removing an `init-param` element) but don't create any `filter-mapping` element.

KerberosLogin_onSuccess

Configure the step to execute after successful authentication. If no step is configured here the process ends and the user will be authenticated.

HostContext_earlyHints

Enables the HTTP/2 feature of early hints.

Configures early hints with the Apache directive H2PushResource

It will send out a "103 Early Hints" response to a client as soon as the server starts processing the request.

UnauthenticatedRealm_sessionTimeout

Defines the idle timeout of a nevisProxy session.

A nevisProxy session will be created only if required (e.g. to store application cookies).

Please set the timeout as low as possible to not increase the risk of session exhaustion attacks.

Nevis recommends not to exceed the default. **A high session timeout in Unauthenticated Realm is strongly discouraged as it opens the door to DoS attacks: nevisProxy can be brought down by creating millions of sessions with simple GET requests.**

FIDO2Onboarding_residentKey

WebAuthn enables high assurance multi-factor authentication with a passwordless login experience. One of the things that enables this is what is called Discoverable Credentials, also referred to as resident keys. This property specifies the extent to which the Relying Party desires to create a client-side discoverable credential.

Allowed values:

- `unspecified`
- `discouraged`
- `preferred`
- `required`

NevisAuthRadiusFacade_secret

Enter a secret to be used for this facade and all Radius clients.

NevisAuthRealm_onDemand

Applications may be configured to trigger a session upgrade flow.

Here you assign the authentication steps which provide these session upgrade flows. This mechanism also works when the realm is accessed via a `SAML IDP`.

The process of selecting and executing a flow is as follows:

An application's `Authorization Policy` specifies the required authentication level (`2 - 9`) which is needed to access the application. (Level `1` is not allowed here, as the session has at least level `1` after the user successfully completes the initial authentication flow.)

Every time the user accesses the application, the policy is enforced as follows:

- If the authentication level of the current session is lower than the level required by the policy, nevisAuth is invoked to execute a session upgrade flow - the one which provides the required level.
- Only if the flow runs through successfully, the level reached is stored in the session and access is granted.
- If the level of the session equals, or is higher than the required level, access is granted immediately.
- Authentication steps assigned here are executed only if the required level (by policy) exactly matches the provided value in its `Authentication Level` property. For example, if level `3` is required, the authentication step directly providing that level is started.

It is possible, in a multi-step flow, that the required authentication is reached only after the second step or later. In this case, assign `Advanced Session Upgrade` as the first step. In this step, you declare the level that should ultimately be reached by the flow. The engine can then match the required level to the one provided by the flow, even if it is not provided by the first authentication step in the flow.

When no flow can be determined the `Default Session Upgrade Flow` will be used instead.

MicrosoftLogin_scope

Select the request scope(s) for getting user information from Microsoft. Default scopes is `email`.

Scope `openid` will be added automatically because Microsoft is implement based on OpenID protocol.

Scope `offline_access` for generate refresh token.

SamlSpConnector_acsUrlWhitelist

By default, the whitelist is calculated based on `SP URL - Assertion Consumer Service(s)`. But in some special cases, you can use wildcards to allow a wide range of whitelisted urls. Examples:

- *.mydomain.com
- mydomain.com*

TANBase_tanFormat

The format of the TAN sent to the user. For instance, with `5 digits`, the generated TAN will always consist of 5 numerical digits (e.g. `12345`).

SAPLogonTicket_systemClient

Identifier of client. See SAP documentation of SAP SSO logon tickets for more information. Default value is SAP's default and should be correct for most cases.

NevisIDMChangePassword_newPassword1

Mandatory input value to use for new password if `Show GUI` is disabled.

OAuth2AuthorizationServer_allowedOrigin

List of URL from where that allow to access the `authorization` endpoint and `token` endpoint. If this field does not set, `authorization` endpoint and `token` endpoint can be access from everywhere.

URLHandler_phase

The phase when this filter should be applied depends on your use case.

- use `START` when the redirect / rewrite should be done as early as possible.
- use `AFTER_SANITATION` to redirect / rewrite after validating the request.
- use `AFTER_AUTHENTICATION` to redirect / rewrite after authentication.

NevisDPDeployable_resourceBase

Configure the base directory for uploaded `Custom Resources`.

Enter a path relative to the instance directory, such as `conf` or `import`.

Absolute paths and nested paths are not supported.

SamlResponseConsumer_samlSigner

Configure the key material for signing outbound SAML messages.

The following messages will be signed: `ArtifactResolve, LogoutRequest, LogoutResponse`

SAML Artifact Binding

To use SAML Artifact Binding with a certain IDP, the `Artifact Resolution Service` must be configured in the `SAML IDP Connector`.

The flow begins with the IDP sending an `ArtifactResponse` message to any of the configured frontend paths.

Now an `ArtifactResolve` message will be created and signed using this certificate. The message will then be sent to the IDP via a server-to-server call.

SamlSpRealm_logoutReminderRedirect

Enter a URL or path to redirect to when a user accesses, and the session has expired.

The redirect is executed on next access in the following cases:

- the user has closed the browser
- user session has expired due to idle timeout

The following requirements must be fulfilled:

- Usage of HTTPs to access the application and for the entire SAML process.
- No other session expiration feature must be used.

SecurosysKeyStoreProvider_certObjectLabel

The certificate objects label on the HSM.

AuthorizationPolicy_level

The `Authentication Level` defines the strength of authentication. Enter a number between `2` and `9` (including).

If the session is not yet at the configured level a session upgrade will be performed.

Level `1` is the weakest possible authentication. By definition this level is reached by the initial authentication flow, e.g. set by a username / password authentication step (e.g. `LDAP Login`).

Level `2` is the default level set by steps which do second factor authentication (e.g. `Test TAN`).

Levels `3` to `9` are not used by default. These levels may be used for additional session upgrade processes.

For the session upgrade to succeed there must be a step which set at least this level. This step must be assigned to `Session Upgrade Flow(s)` in the `Authentication Realm` pattern.

In case the upgrade flow consists of multiple steps and the level should be reached by a subsequent step assign the `Advanced Session Upgrade` pattern instead.

The authentication level defined for an application can be overridden for a sub-path by combining several `Authorization Policy` patterns for this application. The authentication level can also be inherited between patterns. See `Authentication Level Mode` for details.

This setting requires assigning an `Authentication Realm` on the application pattern.

Usage examples:

- Enforce an authentication level for an application: use an `Authorization Policy` pattern with the `Authentication Level` to enforce and link it to the application via `Additional Settings`;
- Enforce an authentication level for some sub-paths of an application: use an `Authorization Policy` pattern with the `Authentication Level` to enforce and `Apply only to sub-paths` set to the paths to protect. Link the pattern to the application via `Additional Settings`;
- Enforce some main authentication level for an application and some specific authentication level for some sub-paths: use two `Authorization Policy` patterns, one with the main `Authentication Level` and no sub-path, and one with the specific `Authentication Level` and `Apply only to sub-paths` set to the paths where the specific authentication level should apply. Link both patterns to the application via `Additional Settings`.
- Enforce some main authentication level for an application and disable them for some sub-paths: use two `Authorization Policy` patterns, one with the main `Authentication Level` and no sub-path, and one with no `Authentication Level` and `Apply only to sub-paths` set to the paths where no authentication level should be enforced. Link both patterns to the application via `Additional Settings`.

- Enforce an authentication level for an application and add some required roles for some sub-paths: use two `Authorization Policy` patterns, one with the `Authentication Level` for the application, `Authentication Level Mode` set to `self-contained`, and no sub-path, and the other pattern with no `Authentication Level`, `Authentication Level Mode` set to `inherited`, the `Required Roles` for the subpaths, `Required Roles Mode` set to `self-contained`, and `Apply only to sub-paths` set to the paths where the required roles should be enforced. Link both patterns to the application via `Additional Settings`.

LuaPattern_script

Upload a Lua script which should be invoked for requests and / or responses. The script has to contain one or multiple of the following Lua functions:

- `function inputHeader(request, response)` - called once per request
- `function input(request, response, chunk)` - called once per request body *chunk*
- `function outputHeader(request, response)` - called once per response
- `function output(request, response, chunk)` - called once per response body *chunk*

The uploaded script will be deployed to the nevisProxy host in sub-directory `WEB-INF` using the name of this pattern for the file name to ensure that the file name is unique.

Here is an example Lua script which replaces sensitive information in response bodies:

```
local buf = {}
function output(request, response, chunk)
  if chunk ~= nil then
    table.insert(buf, chunk)
    return nil
  else
    return string.gsub(table.concat(buf), "some-sensitive-data", "*****");
  end
end
```

The following expressions can be used anywhere within the script:

- `${name}` - sanitized name of this pattern
- `${host}` - name of the `Virtual Host` directory
- `${instance}` - name of the `nevisProxy Instance` directory

JWTToken_secret

Enter a shared secret to be used for symmetric algorithms.

This is required for `JWS` because of the `HS256` algorithm.

SamlToken_issuer

Enter the `Issuer` which will be used to create the token.

If nothing is configured the name of the pattern is taken.

OAuth2AuthorizationServer_invalidTokenRequest

Configure the step to execute after error when token request is invalid and token error response is about to be issued.

If no step is configured here the process ends and the error response issued and return to the client.

NevisLogrendDeployable_port

Enter the port the nevisLogrend shall listen on.

NevisFIDO2Database_schemaPassword

The password of the user on behalf of the schema will be created in the database.

NevisIDMProperty_maxLength

Enter `maxLength` for the property definition file.

Defines the maximum length of the property value.

NevisDetectServiceAccessBase_realm

Mandatory setting to enforce authentication.

NevisAuthRealmBase_templateMode

Choose between two options:

- `additive` : files uploaded as `Login Template` will be added on top of the default. Use this option when you want to **add** or **replace** files, but don't want to upload an entire template. There will be less work when you upgrade.
- `complete` : **only** the files uploaded as `Login Template` will be deployed. Use this option when you want to provide the entire template.

NevisDetectMessageQueueDeployable_port

Enter the port on which nevisDetect MQ will listen.

RoleCheck_roles

Enter 1 or multiple roles, 1 role per line.

If the user has **any** of these roles, the flow will continue with `Found` .

If the user has **none** of these roles, the flow continues with `Not Found` instead.

Roles managed in nevisIDM have the format `<application>.<name>` .

Examples:

- `MyApp.Admin`
- `nevisIDM.Root`

NevisAdaptAnalyzerConfig_deviceFingerprintAnalyzer

Used to disable Device Finger creation and analysis.

CookieCustomization_conflictResolution

When multiple `Cookie Customization` patterns are used it happen that a certain cookie is defined as both a `Client Cookie` and as a `Shared Protected Cookie` for the same application.

By default, this conflict is resolved by allowing the cookie to `pass-through` , treating it as a `Client Cookie` .

This behaviour is usually more robust but less secure as the cookie will be accessible in the browser.

Select `protect` to threat the cookie as a `Shared Protected Cookie` instead.

NevisIDMProperty_accessModify

Possible settings:

- `READ_WRITE` : Input is possible for the if previous value was stored.
- `READ_ONLY` : Field is read only.
- `OFF` : Field is not updatable and property is not displayed GUI.

Users with `AccessControl.PropertyAttributeAccessOverride` can edit these field regardless of this settings.

SocialLoginBase_userId

Logged userId will automatically get from social account. But you can change the userId by using this field.

NevisIDMPasswordLogin_customFinalRedirect

Enter a URL, path, or nevisAuth expression which defines where to redirect to after the new password has been set.

NevisKeyboxProvider_slot

A `Slot` is a directory of a nevisKeybox instance.

By default, nevisKeybox is located at `/var/opt/neviskeybox/default/` . If missing please run the following command on the affected target server(s):

```
neviskeybox handover
```

A `Slot` may contain:

- an arbitrary number of key stores (identified by label)
- up to 1 trust store.

AccessRestriction_dbFile

IP geolocation database file for country filtering.

Currently only the `mmdb` format (MaxMind Database) is supported. This is a binary file format.

SamlSpRealm_timeoutRedirect

Enter a URL or path to redirect to after session timeout.

The redirect is executed on next access when the session has expired.

This is different from the `Logout Reminder Redirect` feature which also performs the redirect when the user comes back after closing the browser.

For this feature an additional cookie `Marker_<name>` will be issued. The value will be set to `login` or `logout` depending on the last user action.

The following requirements must be fulfilled:

- Usage of HTTPs to access the application and for the entire SAML process.
- No other session expiration feature must be used.

AzureServiceBus_expiry

Remote Azure Service Bus Queue to which Expiry messages should be sent.

Messages in Expiry Queue are those messages which validTo time has passed without successful receive action and without failing for other reason. For further reference check `NevisIdm Technical documentation > Configuration > Components > Provisioning module > Provisioning providers`.

OutOfBandManagementApp_host

A virtual host assigned will be used to expose services required for `Out-of-band Management Application`.

AuthorizationPolicy_forbiddenRoles

Optional setting to enforce authorization.

Callers must not have any of the specified roles to access.

Forbidden roles defined for an application can be overridden for a sub-path by combining several `Authorization Policy` patterns for this application. Forbidden roles can also be inherited between patterns. See `Forbidden Roles Mode` for details.

This setting requires assigning an `Authentication Realm` on the application pattern.

Usage examples:

- Enforce forbidden roles for an application: use an `Authorization Policy` pattern with the `Forbidden Roles` to enforce and link it to the application via `Additional Settings`;
- Enforce forbidden roles for some sub-paths of an application: use an `Authorization Policy` pattern with the `Forbidden Roles` to enforce and `Apply only to sub-paths` set to the paths to protect. Link the pattern to the application via `Additional Settings`;
- Enforce some main forbidden roles for an application and some specific forbidden roles for some sub-paths: use two `Authorization Policy` patterns, one with the main `Forbidden Roles` and no sub-path, and one with the specific `Forbidden Roles` and `Apply only to sub-paths` set to the paths where the specific forbidden roles should apply. Link both patterns to the application via `Additional Settings`.

- Enforce some main forbidden roles for an application and disable them for some sub-paths: use two `Authorization Policy` patterns, one with the main `Forbidden Roles` and no sub-path, and one with no `Forbidden Roles` and `Apply only to sub-paths` set to the paths where no forbidden roles should be enforced. Link both patterns to the application via `Additional Settings` .
- Enforce some forbidden roles for an application and add an authentication level for some sub-paths: use two `Authorization Policy` patterns, one with the `Forbidden Roles` for the application, `Forbidden Roles Mode` set to `self-contained` , and no sub-path, and the other pattern with no `Forbidden Roles` , `Forbidden Roles Mode` set to `inherited` , the `Authentication Level` for the subpaths, `Authentication Level Mode` set to `self-contained` , and `Apply only to sub-paths` set to the paths where the authentication level should be enforced. Link both patterns to the application via `Additional Settings` .

JWTToken_type

The following types of JWT token are supported:

- `JWS` : JSON Web Signature - using `HS256` or `HS512` algorithm
- `JWE` : JSON Web Encryption - using `RSA-OAEP-256` and `A256GCM` algorithm

Note: in case asymmetric encryption is used, the `x5t#S256` Certificate thumbprint header parameter will automatically be added according to RFC 7515.

PemKeyStoreProvider_keystoreFiles

Upload your key material in PEM format.

File name	Description	Required
`key.pem`	private key	yes
`cert.pem`	own certificate	yes

File name	Description	Required
`ca-chain.pem`	CA chain	when providing a HTTPS endpoint

Examples

How to produce the required files depends on your setup. The following examples use `openssl`.

Generate a private key:

```
openssl genrsa -des3 -out key.pem 2048
```

Generate a certificate signing request (CSR):

```
openssl req -new -key key.pem -out example.csr -subj "/C=CH/O=Example Company/CN=example.com"
```

If this key store is used to provide a HTTPs endpoint, the common name (CN) should contain the domain.

You can now use the CSR to request a certificate from your CA. For testing a self-signed certificate is often sufficient:

```
openssl x509 -signkey key.pem -in example.csr -req -days 365 -out cert.pem
```

Hardening

We recommend to use a *variable* so that you can use **secrets** to protect the content. This example references 2 *nevisAdmin 4 secrets* storing private key and own certificate:

```
  my-variable:
    - inv-res-secret://f370a14a36db9f29763e8dc1#key.pem
  - inv-res-secret://147cc54a5629fadac761ec01#cert.pem
```

When deploying to Kubernetes, the key material may be stored in a *Kubernetes secret* instead. nevisAdmin 4 does not retrieve Kubernetes secrets during generation and thus all key store files **must** be provided. This example uses a Kubernetes secret `my-secret` :

```
my-variable:
  - k8s-secret-file://my-secret:key.pem/
  - k8s-secret-file://my-secret:cert.pem/
  - k8s-secret-file://my-secret:ca-chain.pem/
  - k8s-secret-file://my-secret:keystore.pem/
  - k8s-secret-file://my-secret:keystore.jks/
  - k8s-secret-file://my-secret:keystore.p12/
  - k8s-secret-file://my-secret:keypass/
```

The additional `keystore.*` files contain private key, own certificate, and the CA chain. You can use the Java `keytool` and `openssl` to produce these files.

The `keypass` file must be a script which is executable by `nvbgroup` and prints the passphrase for `keystore.*` and `key.pem` to stdout.

nevisAdmin 4 does not notice when the content of the Kubernetes secret changes. Manual interaction (terminating pods) is required in that case.

FIDO2Authentication_level

Authentication level that is set on success.

DeployableBase_instanceName

Enter the instance name.

If not set, the pattern name will be used as the instance name.

When deploying to Kubernetes, this setting will be ignored and the instance name will be `default`.

GenericSocialLogin_claimsRequest

The claims request parameter. This value is expected to be formatted in JSON and does not accept trailing spaces nor tabs.

AuthCloudBase_loginMessage

You can set an optional custom message for the login confirmation step.

Only one language is supported here.

For example, `New login request for siven.ch`.

AccessTokenConsumer_onMissingToken

Assign a step to continue with when no token was sent.

If nothing is assigned then authentication will fail with an error.

OAuth2Client_idTokenLifetime

Enter a custom lifetime for the ID token.

If not set the value of the `OAuth 2.0 Authorization Server / OpenID Provider` is used.

NevisAuthDeployable_signerTrustStore

Assign a trust store to validate the signature of the **internal** NEVIS SecToken.

This is an advanced setting and it is usually not required to configure this.

If no pattern, an `Automatic Key Store` pattern, or a `PEM Key Store` , is assigned to `Internal SecToken Signer Key Store` , then you **do not** have to configure this. The configuration of nevisAuth will be generated correctly, based on the deployment type and scaling.

Configuration is required in **classic VM deployment**, when this instance is deployed to multiple hosts, **and** the hosts have **different** key material in the `Internal SecToken Signer Key Store` .

SamlSpConnector_url

Enter the *Assertion Consumer Service URL* of the SP.

Enter multiple values if the same SP can be accessed via multiple URLs.

If the SP is provided by a `SAML SP Realm` the URLs are structured as follows:

- scheme, host and port: `Frontend Addresses` of each `Virtual Host` where the `SAML SP Realm` is used.
- path component: `Assertion Consumer Service` of the `SAML SP Realm` .

The URLs are used during SP-initiated SAML authentication to validate incoming SAML requests. The `assertionConsumerServiceURL` attribute of received SAML `AuthnRequest` messages must match one of these URLs.

The first URL is also used for IDP-initiated authentication (property `spURL` of the `IdentityProviderState`).

IDP-initiated authentication may be triggered by sending a request to any of the `Frontend Path(s)` of the `SAML IDP` . The following parameters must be provided either in the query or as `POST` parameters:

- `Issuer` - the unique name used by the SP (also called `entityID` in the SAML metadata).
- `RelayState` - will be sent back to the SP together with the SAML `Response` when authentication is done. In case the SP is setup by a `SAML SP Realm` this should a URL of an application protected by this realm.

SamlSpConnector_audienceRestriction

Set custom audience(s).

If you need multiple `<Audience>` elements in the generated `<AudienceRestriction>` , enter multiple lines.

This configuration is ignored when `Audience Restriction` is set to `issuer` or `none` .

Check the documentation of the service provider on what is expected.

AuthServiceBase_path

Enter frontend path(s) which should be handled.

MobileTAN_channel

The connection provider for the TAN transmission. Currently the only supported connection provider is a SwissPhone SMS Gateway.

NevisDPLogSettings_serverLogFormat

Log4j 2 log format for the default SERVER logs.

Note: not relevant when Log Targets is set to `syslog` .

SAPLogonTicket_caching

If set to `enabled` , this property enables the `CachingAllowed` flag in the issued ticket. See SAP documentation of SAP SSO logon tickets for more information.

NevisAdaptRememberMeConnectorStep_fingerprintJsVersion

This configuration option gives the administrator the ability to ensure backwards compatibility in case so far V2 fingerprints have been in use.

- `V2` - to ensure backward compatibility, FingerprintJS V2 will be used
- `V3` - default option, uses FingerprintJS V3

AccessTokenConsumer_authorizationServer

Assign the `OAuth 2.0 Authorization Server / OpenID Provider` which has issued the access token.

Note that this step works in combination with Nevis `OAuth 2.0 Authorization Server / OpenID Provider` only and the other pattern has to be in the same project.

NevisMetaDeployable_properties

Configure properties of the nevisMeta.

Add or **overwrite** properties by entering a value.

Remove properties generated by this pattern by leaving the value empty.

Examples:

Key	Value
database.migration.automatic	false

SamlSpRealm_stepupPath

Applications may redirect to this location to force the SP to invoke the IDP again by sending an `AuthnRequest` .

This mechanism allows applications to enforce a *session upgrade*.

The URL must contain the following query parameters:

- `relayState` : the path to redirect to after successful session upgrade.
- `level` : the required authentication level (`2-9`). The level will be sent to the IDP within the `RequestedAuthnContext` .

Tokens produced by `Application Access Token` patterns assigned to applications will be re-created on next access to reflect updated user data.

Example for `RequestedAuthnContext` with `level=2` :

```
<saml2p:RequestedAuthnContext>
  <saml2:AuthnContextClassRef xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">urn:nevis:level:2</saml2:AuthnContextClassRe
</saml2p:RequestedAuthnContext>
```

NevisIDMPasswordLogin_emailRedirPathValidationMode

Defines how to validate the redirection path which sent in the password reset e-mail. The following modes are available:

- `Allow-list regexes` : Only paths that match the regexes are allowed. Only paths needs to be defined (For example in case of `https://your-domain.com/your-path` , only /your-path needs to be defined). Regexes can be defined in `Custom Redirection Path Validation Regexes` .
- `Deny-list Regexes` : All paths are allowed except those that match the regexes. Only paths needs to be defined (For example in case of `https://your-domain.com/your-path` , only /your-path needs to be defined). Regexes can be defined in `Custom Redirection Path Validation Regexes` .

TransformVariablesStep_variables

Set variables.

To find out which variables are available when a request comes in set the log level of `Vars` to `DEBUG` and check the nevisAuth log.

The following syntax variants are supported:

```
<scope>:<name> =
<scope>:<name> = some value
<scope>:<name> = ${some-auth-expression}
<scope>:<name> = #{some-EL-expression}
```

The setting `On Empty Value` defines how null values and empty Strings shall be handled.

Example: store the query parameter `RelayState` in the session:

```
sess:RelayState = ${inargs:RelayState}
```

Example: clear *finishers* registered in the session:

```
sess:ch.nevis.session.finishers =
```

If you want to use advanced features of the `TransformAttributes` state, provide the required configuration via `Custom Properties` .

DatabaseBase_encryption

If `enabled` the query parameter `useSSL=true` will be added to enable 1-way TLS.

If no `Trust Store` is assigned then `trustServerCertificate=true` will be added to the connection string.

Assignment of a `Trust Store` is recommended for production use.

Note: `PostgresSQL` database connection configuration doesn't support TLS connection yet.

ErrorHandler_mode

Enable or disable the error handling.

When set to `disabled` , all settings except `Apply only to sub-paths` are ignored. Use this setting in combination with `Apply only to sub-paths` to disable the error handling for some sub-paths only.

Usage examples (valid for `Virtual Host` s and backend applications):

- Disable the error handling: use an `Error Handler` pattern with `Mode` set to `disabled` and link it to the target pattern via `Additional Settings` ;
- Disable the error handling for some sub-paths: use an `Error Handler` pattern with `Mode` set to `disabled` and `Apply only to sub-paths` set to the paths where no error handling should occur, and link it to the target pattern via `Additional Settings` ;
- Define a customised error handling and disable it for some sub-paths: use two `Error Handler` patterns, one with the custom settings, and one with `Mode` set to `disabled` and `Apply only to sub-paths` set to the paths where no error handling should occur. Link both of them to

the target pattern via `Additional Settings` .

DatabaseBase_user

Database connection user.

This setting is used in the following cases:

- Classic deployments (VM)
- In Kubernetes when 'Database Management' (Advanced Settings) is set to 'disabled'.

GenericSocialLogin_buttonLogo

The path to logo file of the social login provider. This path is the path of logo file which you which uploaded at Login template in Realm pattern. E.g:

In the zip file the icon with path `/webdata/resources/icons/icon.csv` , the input is `/icons/icon.csv`

NevisAdaptDeployable_ipReputationUpdateURL

Provide a download URL for the database file. The file is downloaded then moved over to the path defined above.

OAuth2Client_grantTypes

Enter the allowed grant types.

AppleLogin_redirectURI

The callback URI to go to after a successful login with Apple.

This will create an endpoint in your host config.

The URL will be a combination of the `Frontend Address` of the `Virtual Host` and the value configured here. For example, let's assume that you have configured:

- Return Path: `/oidc/apple/`
- Frontend Address: `https://nevis.net`

Then the URL will be `https://nevis.net/oidc/apple/` .

Use the `exact:` prefix to use the given path as-is. Without this prefix a normal mapping with `/*` will be generated and thus sub-paths will be accessible as well.

CustomAuthLogFile_serverSyslogFormat

Log4j 2 log format for the SERVER SYS logs.

Note: not relevant when Log Targets is set to `default` .

NevisAdaptDeployable_ipPrivateNetworkFilter

If your network you are connecting from only contains private network addresses nevisAdapt will filter these addresses out thus not assigning riskscore to GeoIP data from these addresses.

If you wish to disable this feature, you can do so by setting the following.

```
nevisAdapt.database.ipPrivateNetworkFilter.enabled=false
```

The default value is `true` .

GenericIngressSettings_annotations

Add Kubernetes annotations to customize the behaviour of the NGINX ingress.

Restrict access based on source IP:

```
nginx.ingress.kubernetes.io/whitelist-source-range: 213.189.148.0/24,173.245.48.0/20,103.21.244.0/22
```

Increase the maximum allowed request size:

```
nginx.ingress.kubernetes.io/proxy-body-size: 10m
```

Please read Annotations - NGINX Ingress Controller for details.

SocialLoginBase_host

Assign a `Virtual Host` which shall serve as entry point for the callback from social login provider.

In case your host has

- 1 address, that address will be used
- many addresses with
  - 1 https, and many http, the https will be used without warning

- mix between http and https, the 1st https will be used with warning
- single scheme (http or https only) the 1st address will be used with warning

E.g.

```
http://nevis.net
http://nevis-security.net
https://nevis.net
https://nevis-security.net
```

The `https://nevis.net` will be used as the host for Apple callback

SAPLogonTicket_encoding

Encoding to use for the SAP token. Note that some SAP applications (in particular those running as native processes) do not support all encodings. In such cases, error messages may be misleading. Usage of the encoding "ISO8859-1 (ISO-LATIN-1)" is encouraged as this seems to be supported by all SAP products.

The default is `ISO8859-1` .

ResponseRewritingSettings_responseBody

Configure response body rewrite rules.

In the first column enter a regular expression. In the second column enter the replacement.

Rules will be applied to each line of the response body.

Response body rewriting can be a complex task and should only be done if there is no other way. Use the browser's network tracing to have a look at the responses to find out what needs to be rewritten.

Examples:

Regex	Replacement	Description
`http://my-backend.intra.siven.ch`	`https://www.siven.ch`	replace an internal host name with the external one
`https?://[^/]+(/.*)`	`$1`	make links relative
`<base href="/">`	`<base href="/app/">`	apps which have a context root of `/` may require a rewrite of the `base` element

For further information see documentation of `RewriteFilter` in nevisProxy Technical Documentation.

RuleBundle_whitelistRules

Configure *whitelist modifications*.

As explained in the ModSecurity documentation *whitelist modifications* are applied **before** including the core rules.

If both the `Request Validation Settings` and the `Rule Bundle` pattern have *whitelist modifications* configured, first the `Rule Bundle` , then the `Request Validation Settings` whitelists will be applied.

Note that new rule may require a rule ID which has to be unique for this pattern. Use the range 1-99,999 as it is reserved for local (internal) use.

- Remove rule with ID `900200` for the path `/app/some.html` :

```
SecRule REQUEST_URI "@streq /app/some.html" "pass,nolog,id:1000,ctl:ruleRemoveById=200002"
```

OAuth2AuthorizationServer_invalidAuthorizationRequest

Configure the step to execute after error when the authorization request is invalid. Example:

- Cannot parse Authorization Request
- Request `response-type` mismatch with client configuration
- Invalid scope
- Policy not allow
- PKCE method not support
- Missing code challenge
- Plain code challenge

If no step is configured here the process ends and the error will display on UI.

NevisAuthDeployable_resources

Upload additional resources required by your configuration.

Uploaded files will be deployed into the `conf` folder of the nevisAuth instance.

GenericAuthenticationStep_authStatesFile

Upload an XML file containing `AuthState` elements.

Example to illustrate the syntax:

```
<AuthState
  name="${state.entry}"
  class="ch.nevis.esauth.auth.states.standard.ThrottleSessionsState"
  final="false">
  <ResultCond name="ok" next="${state.done}" />
  <Response value="AUTH_ERROR">
    <Gui name="AuthErrorDialog"/>
  </Response>
  <property name="queryValue" value="${request:userId}" />
</AuthState>
```

See Standard authentication AuthStates and plug-ins for further examples.

The following expressions may be used:

- `${instance}` : name of the nevisAuth instance.
- `${request_url}` : generates a nevisAuth expression which returns the URL of the current request
- `${realm}` : name of the Realm (see below)
- `${state.entry}` : use as `name` to mark the first `AuthState` .
- `${state.done}` : use as `next` in `ResultCond` elements to exit this step and continue with `On Success` .
- `${state.failed}` : use as `next` in `ResultCond` elements to exit this step and continue with `On Failure` .
- `${state.exit.<index>}` : use as `next` in `ResultCond` elements to exit this step and continue with an `Additional Follow-up Step(s)` . The index starts with `1` .
- `${state.level}` : must be used if an `Authentication Level` has been defined. Use as `authLevel` on `ResultCond` elements which point to `${state.done}` .
- `${keystore}` : name of the `KeyStore` element provided by this pattern. Assign a pattern to `Key Objects` to add a `KeyObject` into this `KeyStore` .
- `${service.postfix}` : in Kubernetes side-by-side deployment a postfix is added to service names. Use this expression when connecting to a service deployed against the same inventory.

- `${var.<name>}` : insert the scalar variable `<name>` . This is an alternative to using `Template Parameters` .

The `name` of `AuthState` elements is prefixed with the sanitized name of the Realm (referred to as `${realm}`).

The realm prefix must be added when using `propertyRef` to reference AuthStates generated by other patterns (e.g. `<propertyRef name="${realm}_SomeState"/>`).

An exception is the add-on pattern `nevisIDM Connector for Generic Authentication` which does not set a prefix. Here the `propertyRef` must be defined as follows:

```
<propertyRef name="nevisIDM_Connector"/>
```

This pattern does not validate that labels are translated. Translations can be provided on the `Authentication Realm` pattern.

TANBase_onSuccess

Configure the step to execute after successful authentication. If no step is configured here the process ends and the user will be authenticated.

OutOfBandMobileAuthentication_onFailure

When authentication fails due to user behaviour, the authentication flow may continue with assigned step.

The authentication may fail due to the following reasons (non-exhaustive list):

- A timeout has occurred
- The authentication itself has failed (for example wrong biometric credential was provided)
- Client errors (e.g. the authenticator chosen did not comply with the policy)

To handle a failure upon sending a push notification, configure `On Push Failure` instead.

NevisIDMProperty_clientExtId

Enter `clientExtId` for the property definition file.

If set, the property becomes specific to the referred client. Otherwise, the property is client-independent.

NevisProxyDeployable_cacheTimeout

Configures the number of seconds before an SSL session expires in the SSL Session Cache.

For more information, see the documentation of the SSLSessionCacheTimeout directive.

NevisAuthRealmBase_secRoleParams

Add custom `init-param` elements to the `SecurityRoleFilter` generated by this pattern.

Multi-line values, as required for conditional configuration, can be entered by replacing the line-breaks with `\n` .

NevisLogrendLogSettings_regexFilter

If set, messages for `nevislogrend.log` which match the given regular expression won't be logged.

The regular expression must match the entire line. For instance, you may use the following format to match `some text` :

```
.*some text.*
```

Example: drop messages caused by the Kubernetes liveness checks

```
.*GET /nevislogrend/health.*
```

KeyObject_keyObjectId

Set the attribute `id` of the `KeyObject` element.

The `id` must be unique within the nevisAuth instance. If not set the sanitized name of this pattern will be used.

NevisAdaptDeployableBase_bindHost

Enter a custom host name to listen on.

This setting is relevant in classic VM deployment, when working with multi-homed target hosts.

In Kubernetes the component listens on `0.0.0.0` and thus this setting is discouraged.

JWTAccessRestriction_algorithm

The algorithm used to sign and verify the JWT.

Supported algorithms are:

- RS256
- RS384
- RS512 (default)

NevisFIDODeployable_backendKeyStore

The key nevisFIDO uses to connect to `nevisIDM Instance` . Important to note that the certificate that belongs to this key must exist in nevisIDM as the certificate credential of the nevisfido technical user.

NevisIDMGenericBatchJob_job

Add configuration of a bean which configures your batch job.

The basic syntax is as follows:

```
<bean id="someJobId" class="org.springframework.scheduling.quartz.JobDetailFactoryBean">
    <property name="description" value="Some job description"/>
    <property name="durability" value="true"/>
    <property name="jobClass" value="some.job.Class"/>
    <property name="jobDataMap">
        <bean class="org.quartz.JobDataMap">
            <constructor-arg>
                <map>
                    <entry key="someJobParam" value="some value"/>
                </map>
            </constructor-arg>
        </bean>
    </property>
</bean>
```

NevisAuthRealmBase_defaultProperties

Add or overwrite properties in the `default.properties` of the nevisLogrend application.

Use only when there is no high-level setting. We recommend to **not** overwrite any language related properties, as the languages should be in sync with nevisAuth. You can configure the languages on the `nevisAuth Instance`.

Requires that nevisLogrend is used for GUI rendering. Check the help of `Login Renderer` for details.

SAPLogonTicket_systemId

Identifier of issuing system (or issuer). This must match the key under which the issuer certificate was configured in the consuming service.