NevisIDMPasswordLogin_passwordResetEnabled

Enables the password reset process.

The password reset process works as follows:

- The user has to enter his login ID or email.
- An email with a link be sent to the user.
- The user has to click the link in the mail to set a new password.

A link will be added to the login page. Users may click this link if they have forgotten their password to request a new password.

The link text can be changed on the Realm pattern by setting translations for the label `pwreset.info.linktext` .

NevisFIDO2Database_maxConnectionLifetime

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

NevisIDMChangePassword_displayPasswordPolicy

If `enabled` , the active password policy is displayed on the GUI.

NevisFIDODeployable_policy

A FIDO UAF Policy defines which authenticators can be used during registration or authentication.

Each configuration file has to contain exactly 1 policy, as a JSON object.

A policy contains a list of allowed and/or disallowed AAIDs, pointing to specific authenticator implementations.

The policy for a certain operation can be specified in the `context` of the GetUafRequest object.

Example Javascript snippet for a registration operation with `pin_only` policy:

```
const getUafRequest = {
    op: "Reg",
    context: JSON.stringify({username: userExtId, policy: "pin_only"})
}
```

Given a policy in the `GetUafRequest` context, nevisFIDO looks for a `.json` file with the same name.

There must be at least one file named `default.json` which serves as default policy. This policy will be used when no policy is specified in the `GetUafRequest` context.

The following policies are included by default:

- `default` : allows to execute registration or authentication with all authenticators supported by Nevis.
- `pin_only` : allows only the PIN authenticators.
- `biometrics_only` : allows only biometric authenticators.
- `password_only` : allows only (alphanumeric) password authenticators.

:::info supported authenticators Note that two authenticators supplied by the policy files are currently only supported by the Nevis Mobile Authentication SDK but not the Nevis Access App, these authenticators are:

- The Device Passcode authenticator
- The Password authenticator :::

NevisMetaRESTServiceAccess_meta

Reference the nevisMeta Instance.

Webhook_headers

Configure headers to send along with the call.

NevisIDMChangePassword_onSuccess

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

4.13.0

Full changelog:

Patterns 4.13.0 Release Notes - 2021-11-17

ModSecurity Core Rule Set Upgrade

The *OWASP ModSecurity Core Rule Set* (CRS) version has been upgraded from `3.0.2` to `3.3.2`.

To choose between the CRS versions, a new setting `OWASP ModSecurity CRS Version` has been added to the `Virtual Host` pattern under the `Security` tab.

Please note that Nevis cannot ensure that the new rules are compatible. If you cannot afford testing your applications you can select the previous version.

If you have uploaded a custom rule set (`ModSecurity Rule Set` in the `Virtual Host` pattern), you have to select `custom`.

Changes to TLS Settings

Updated the `compatible` configuration for the `Frontend TLS Settings` of Virtual Hosts. Please refer to the Help for the new values.

Blank fields in `TLS Settings` patterns assigned to a Virtual Host will be now be replaced by the corresponding `recommended` value. The `compatible` value was previously applied.

Fixed an issue where a Virtual Host could have `Frontend TLS Settings` set to `recommended` or `compatible`, and have a `TLS Settings` pattern assigned at the same time. Now assigning a `TLS Settings` pattern requires setting `Frontend TLS Settings` to `custom`.

JWTAccessRestriction_publicKey

The public key corresponding to the private key which was used to sign the JWT.

SamlSpConnector_keyInfo

The `KeyInfo` embedded into the signature informs the service provider about the signer credential used.

Enter one or several of the following elements:

- SKI
- Certificate
- CertificateChain
- Subject
- IssuerSerial
- CRLs
- SubjectDNAsKeyName
- SubjectCNAsKeyName
- KeyNames
- EntityID

- PublicKey

Note that only configured fields defined in the signer certificate are actually added to the `KeyInfo` structure.

RequestValidationSettings_parameters

Define *parameters* which may be used within rules.

Enter a map of key-value pairs.

For instance, a parameter `my_param` could be defined as follows:

```
my_param: 900200
```

These parameters can be used in:

- `Additional Rules`
- `Whitelist Modifications`
- `Exception Modifications`

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 `\}`.

In case a variable `my-variable` is used this is the format you need to use in the inventory:

```
my-variable: |
  my_param: 900200
```

GenericSocialLogin_firstNameClaim

The claim that contains the first name of the logged-in user in the social account. The default value is `given_name`.

PropertiesTestPattern_pathProperty

Enter a path starting with `/`. Usually this is then used to create a mapping in nevisProxy ending with `/*`. Supports the special prefix `exact:` to declare that an exact mapping is desired instead of a nested on in nevisProxy.

GenericAuthXmlServiceBase_parameters

Define *Template Parameters*.

Examples:

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

These parameters can be used in your `Configuration`.

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 `\}` .

SamlSpRealm_idp

Assign a `SAML IDP Connector` for each SAML Identity Provider.

SP-initiated authentication with multiple IDPs requires a `Selection Expression` to be configured for each connector.

NevisAdaptFeedbackConfig_feedbackRedirectURL

Provide a URL to redirect to after sending a report by pressing the feedback link in the notification. This can either be a base homepage or a more security-oriented one (for example page for password reset).

If it remains unset, a basic informative text is displayed about the report instead of a redirect.

SamlIdp_preProcess

An an authentication step to execute before dispatching according to the issuer.

NevisAuthDatabase_synchronizeSessions

Defines when sessions are stored to the remote session store.

- `after-successful-authentication`

A session is stored to the remote session store once authentication has been completed successfully.

Use this mode when performance is critical, when the authentication flow is stateless (e.g. no GUI is shown), when there is only 1 nevisAuth instance, or in classic VM-based deployment scenarios.

- `always`

Session are always stored to the remote session store.

This mode requires nevisAuth version 4.28.0.216 or newer.

We recommend this mode when you want to restart nevisAuth without user impact and for deployments to Kubernetes.

Typically, while a user is logging in, multiple requests are sent to nevisAuth.

If you configure multiple replicas of nevisAuth, the Kubernetes service may forward these requests to any of the replicas. By selecting `always`, you ensure that all replicas can access the user's session during the authentication process.

- `recommended (default)`

Uses `always` when deploying to Kubernetes and `after-successful-authentication` for classic, VM-based deployments.

ErrorHandler_contentTypeMode

The **Content-Type Mode** allows enabling or disabling the error handling depending on the `Content-Type` header of the backend response. Use this setting in combination with the **Content-Types** setting.

Choose one of:

- `None` : The error handling settings are applied to all backend responses.
- `Enabled` : The error handling settings are enabled only for the backend responses with a `Content-Type` header included in the **Content-Types** setting. Backend responses with other Content-Types are propagated to the client.
- `Disabled` : The error handling settings are disabled for the backend responses with a `Content-Type` header included in the **Content-Types** setting. These responses are propagated to the client. The error handling settings are applied to backend responses with other `Content-Type` headers.

GenericAuthenticationStep_onFailure

Use `${state.failed}` to continue with the assigned step.

If no step is assigned and `${state.failed}` is used an `AuthState` named `<Realm>_Authentication_Failed` is generated.

AccessRestriction_subPaths

Set to apply this pattern on some sub-paths only.

Sub-paths must be relative (e.g. not starting with `/`) and will be appended to the frontend path(s) of the virtual host (`/`) or applications this pattern is assigned to.

Sub-paths ending with `/` are treated as a prefix, otherwise an exact filter-mapping will be created.

The following table provides examples to illustrate the behaviour:

Frontend Path	Sub-Path	Effective Filter Mapping
/	secure/	/secure/*
/	accounts	/accounts
/	api/secure/	/api/secure/*
/	api/accounts	/api/accounts
/app/	secure/	/app/secure/*
/app/	accounts	/app/accounts
/app/	api/secure/	/app/api/secure/*
/app/	api/accounts	/app/api/accounts

EmailTAN_recipient

Enter a nevisAuth or EL expression for the recipient.

You have to ensure that this expression always resolves. There will be a system error if the expression does not produce an email address.

Examples:

```
${sess:ch.nevis.idm.User.email}
```

NevisAdaptAnalyzerConfig_geoAnalyzer

Used to disable GeoLocation analysis. If you wish to disable this setting also consider disabling the IP Geolocation settings as well in the `nevisAdapt Instance / IP Geolocation` configuration and the `nevisAdapt Instance / IP Reputation` configuration.

NevisConnectorPattern_kubernetes

This setting is used when deploying to Kubernetes only.

Choose between:

- `disabled` : instance running on a VM.
- `same_namespace` : service running in the same cluster and namespace.
- `other_namespace` : service running in the same cluster but in another namespace.
- `other_cluster` : service running in another cluster.

NevisAuthRealmBase_logrendTrustStore

If nevisLogrend is used and the connection to nevisLogrend uses HTTPs then a trust store should be configured here.

If no pattern is assigned the nevisAdmin 4 automatic key management will set up a trust store.

Maintenance_end

Enter the end date and time of the maintenance window.

- format: `yyyy-mm-dd HH:mm` (24 hours)
- timezone: UTC (not your local time)

- example: `2020-05-20 15:00`

NevisFIDODeployable_basicFullAttestation

Android Key Attestation relies on FIDO UAF Full Basic Attestation in the backend.

Whether Android Key Attestation is applied is controlled by the FIDO UAF policy.

Here you can choose between 2 levels for the FIDO UAF Full Basic Attestation:

`default` - does FIDO UAF Basic Full attestation solely based on the UAF protocol specification. In practice, this means it's limited to verifying the certificate chain.

`strict` - is doing additional checks on the attestation extension of the `TBSCertificate`, ensuring that all key material is kept in certified secure hardware and the phone bootloader was not manipulated.

Additional information can be found in the nevisFIDO UAF configuration documentation including the specifics of the additional checks done in `strict` mode

Refer to the following sections for additional information of how to create the necessary policy files:

- nevisFIDO policy configuration
- support of basic full and basic surrogate attestations

BackendServiceAccessBase_sessionTermination

Use this feature to terminate sessions on the backend application.

nevisProxy will send a `GET` request to this path when the nevisProxy session is terminated (due to logout or session timeout).

JSONResponse_type

Use `AUTH_CONTINUE` to keep the current session and stay in state. When the next request comes in, the `On Continue` exit will be taken.

Use `AUTH_DONE` to complete the current flow and establish an authenticated session. The request will continue in the filter chain in nevisProxy, towards a backend application.

Use `AUTH_ERROR` to terminate the flow, removing the session. Note that this type may also be used for successful execution, to remove the session.

SecurosysKeyStoreProvider_keyObjectLabel

The key objects label on the HSM.

LdapLogin_connectionPassword

Password of the connection user. The user is part of the LDAP connection url.

Example:

- secret://Ll41Zsw54rmeNi2ZeoZD
- verySecretPassword

See the nevisAuth Reference Guide `UseridPasswordAuthenticateState` for more details on how to use obfuscated password.

NevisAdaptDeployable_sharedStorageSettings

Configure this to override the default configurations used for the shared storage in Kubernetes deployments. If you would use an existing shared volume please only set the claim name. This storage should support the ReadWriteMany access mode.

For more information regarding persistent volumes in Kubernetes please visit this page

EmailTAN_onFailure

Assign the step to execute in case no TAN code can be sent or all attempts had been exhausted.

The step will be executed in the following cases:

- the `Recipient` could not be determined
- all attempts had been exhausted and the user has failed to authenticate

If no step is assigned then the authentication flow will be terminated and an error GUI with label `error_99 ( System Problems )` will be shown.

SamlIdpConnector_authRequestLifetime

SAML authentication requests have a maximum lifetime which may be validated by the identity provider.

The lifetime should be low but high enough so that the authentication works on slow network connections.

SAPLogonTicket_keystore

Assign a pattern which sets the key material used for signing the token.

If no pattern is assigned automatic key management is used and the signer key will be created automatically.

AuthorizationPolicy_levelMode

The `Authentication Level Mode` defines which `Authentication Level` is set for the current paths.

When combining several `Authorization Policy` patterns for an application, this setting allow inheriting the `Authorization Level` from a more general pattern.

Choose one of:

- `self-contained` : The `Authentication Level` defined in this pattern is applied to the current paths. They override any `Authentication Level` set on parent paths. If no `Authentication Level` is set in the current pattern, no authentication level will be enforced for the current paths.
- `inherited` : The `Authentication Level` in this pattern is not used. Use this setting if you have another `Authorization Policy` pattern applied to a parent path to inherit the configuration from. For the `Authentication Level` to be inherited from a particular parent, this setting has to be set to `default (self-contained)` in the parent pattern (otherwise you may inherit a value from a grandparent).

FIDO2Onboarding_displayName

Enter a 1 line Groovy statement to determine the `displayName` included in the call to the Registration Options Service.

The statement must produce a String.

The `displayName` is required by nevisFIDO and may be shown to the user by some devices.

Examples:

```
"${session['ch.nevis.idm.User.firstName']}_${session['ch.nevis.idm.User.name']}"
```

DatabaseBase_rootCredential

Enter the name of a Kubernetes secret which contains the user and password of a database root account.

Required in Kubernetes deployment when `Advanced Settings / Database Management` is to `complete` or `schema`.

This is the default behaviour in Kubernetes.

With `complete` the secret should contain the following:

```
username: <root-user
password: <root-password>
```

If the `Database Management` is set to `schema` the root user can be omitted, but the application and schema user has to be specified:

```
ownerUsername: <some-username>
ownerPassword: <some-password>
appUsername: <some-username>
appPassword: <some-password>
```

If used with `complete` the app and owner users will be created with the credentials specified in the secret.

Due to the usage of schemas, it is recommended to create a separate Kubernetes secret for each database pattern with the app and owner credentials when using Oracle or PostgreSQL.

NevisDetectRiskPluginBase_url

Service URL used to connect to the plugin

OutOfBandMobileAuthentication_username

The `username` is used by nevisFIDO to look up the user in nevisIDM.

Depending on how the `nevisFIDO FIDO UAF Instance` is configured, either the `extId` or the `loginId` have to be used.

NevisDetectMessageQueueDeployable_messageBrokerName

The name for the broker to configure ActiveMQ with.

GoogleLogin_buttonLabel

Enter the text that should be displayed for the end-user on the social login button, and provide translations for this label on the Authentication Realms.

InBandMobileAuthenticationRealm_trustStore

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

NevisMetaDeployable_addons

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

OutOfBandMobileStepBase_onCancel

Assign an authentication step to continue with when the user clicks cancel.

Use to provide a fallback authentication option.

You can change the text on the cancel button by translating the label `mobile_auth.cancel.button.label`.

SamlIdpConnector_properties

Enter custom properties for the nevisAuth `ServiceProviderState`.

Example: overwrite `authnContextClassRef` in the `AuthnRequest`

```
out.authnContextClassRef = urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified
```

Example: remove `authnContextClassRef` from `AuthnRequest`

```
out.authnContextClassRef =
```

OAuth2AuthorizationServer_propagationScope

Define propagation scope to store information for following AuthStates. Following are propagated data:

- Authorization request:
  - oauth2.authorization_request.[requestParameter]
- Client configuration:
  - oauth2.client.id
  - oauth2.client.metadata.[field]

- Scope configuration:
  - oauth2.scope.policy.clientCredentialsFlow
  - oauth2.scope.policy.authorizationCodeFlow
  - oauth2.scope.policy.implicitFlow
  - oauth2.scope.policy.refreshTokenRequest
  - oauth2.scope.policy.authenticationRequired
  - oauth2.scope.metadata.[field]

You can find out more information in the output session of AuthorizationServer

The propagated data is intended for logging purposes only, no standard AuthState will use it. But you can access and use with `Groovy Script Step`.

Note: If your flow has multiple user interactions, use the scope `session` to ensure that the information is available throughout the flow.

LdapLogin_type

Configure the type of LDAP directory.

SamlIdp_samlSigner

Configure the key used by this Identity Provider to sign outgoing SAML Assertions.

ErrorHandler_subPaths

Set to apply the error handling on some sub-paths only.

Sub-paths must be relative (e.g. not starting with `/`) and will be appended to the frontend path(s) of the virtual host (`/`) or applications this pattern is assigned to.

Sub-paths ending with `/` are treated as a prefix, otherwise an exact filter-mapping will be created.

The following table provides examples to illustrate the behaviour:

Frontend Path	Sub-Path	Effective Filter Mapping
`/`	`secure/`	`/secure/*`
`/`	`accounts`	`/accounts`
`/`	`api/secure/`	`/api/secure/*`
`/`	`api/accounts`	`/api/accounts`
`/app/`	`secure/`	`/app/secure/*`
`/app/`	`accounts`	`/app/accounts`
`/app/`	`api/secure/`	`/app/api/secure/*`
`/app/`	`api/accounts`	`/app/api/accounts`

ObservabilityBase_parameters

Provide parameters for your configuration file.

Examples:

```
connectionString = InstrumentationKey=00000000-0000-0000-0000-000000000000
```

```
tracesEndpoint = https://otel-collector:4318/v1/traces
metricsEndpoint = https://otel-collector:4318/v1/metrics
logsEndpoint = https://otel-collector:4318/v1/logs
```

ErrorHandler_blockedStatusCodes

Hide certain HTTP status code(s) by returning 200 OK instead (by using the reset-status-code action).

By default, the status code is not changed as it can be useful for technical clients.

The response body will still be replaced.

You may also enter:

- ranges of status codes (e.g. `500-599`),
- lists (e.g. `403,500`)
- combination thereof (e.g. `403,500-599`).

GenericAuthRealm_dependencies

In case your `AuthState` elements use custom classes upload the required JAR file(s) here.

Files uploaded here will be deployed into the `lib` directory of the nevisAuth instance.

OAuth2UserInfo_realm

Assign a realm which shall be exposed to get user information of an OAuth2 Authorization Server or OpenID Connect Provider.

NevisIDMClient_clientName

The name of the client.

DummyTAN_onSuccess

Set the next step on successful entry of the TAN code. If no step is assigned, the process ends and the user will be authenticated.

NevisIDMUserUpdate_mandatoryAttributes

Define which attributes are required and how to provide them.

Example:

```
clientExtId: 100
email: ${sess:email}
remarks:
mobile:
```

NevisAuthDeployable_port

Port the nevisAuth instance is listening on.

HostContext_sessionResource

Exposes the REST interface of the session store servlet on the given path. For security reasons, only DELETE requests are allowed and assigning an access restriction pattern is recommended.

Before setting this parameter, make sure that there will be an actual session store servlet.

KerberosLogin_limitSessionLifetime

If set to `true` then the lifetime of the underlying Kerberos service ticket used by the client during the SPNEGO negotiation will be considered when determining the lifetime of Nevis session. In this case the expiration time of Nevis session cannot be longer than the expiration time of the Kerberos service ticket.

Default is `false`.

GenericAuthenticationStep_onSuccess

Use `${state.done}` to continue with the assigned step.

If no step is assigned and `${state.done}` is found an `AuthState` named `<Realm>_Prepare_Done` will be used instead.

StaticContentCache_subPaths

Set to apply the cache handling on some sub-paths only.

Sub-paths must be relative (e.g. not starting with `/`) and will be appended to the frontend path(s) of the virtual host (`/`) or applications this pattern is assigned to.

Sub-paths ending with `/` are treated as a prefix, otherwise an exact filter-mapping will be created.

The following table provides examples to illustrate the behaviour:

Frontend Path	Sub-Path	Effective Filter Mapping
/	secure/	/secure/*
/	accounts	/accounts
/	api/secure/	/api/secure/*
/	api/accounts	/api/accounts
/app/	secure/	/app/secure/*
/app/	accounts	/app/accounts
/app/	api/secure/	/app/api/secure/*
/app/	api/accounts	/app/api/accounts

GenericIngressSettings_clientCertAuth

Enables client certificate validation in the NGINX Ingress.

Please note that client cert validation cannot be used when the TLS connection is terminated in front of the NGINX Ingress.

Choose between:

- `enabled` : Request a client certificate that must be signed by a certificate that is included in the `CA Secret` . Failed certificate verification will result in a status code `400 (Bad Request)` (unless `Error Page` is configured`).
- `optional` : Do optional client certificate validation against the CAs. Requests will fail with status code `400 (Bad Request)` when a certificate is provided that is not signed by the CA (unless `Error Page` is configured`). When no or an otherwise invalid certificate is provided, the request does not fail, but instead the request is allowed to pass through.

- `optional_no_ca` : Do optional client certificate validation, but do not fail the request when the client certificate is not signed by the CAs from `CA Secret` . The secret still has to exist with a valid certificate.
- `disabled` (default): Don't request client certificates and don't do client certificate verification.

This setting is used to generate the following annotation for the NGINX Ingress:

```
nginx.ingress.kubernetes.io/auth-tls-verify-client
```

GenericHostContextSettings_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 `/*` .
- `both` : like `automatic` but additional `filter-mapping` elements are allowed as well.

SamlSpRealm_timeoutPage

Renders a timeout page when the user session has expired.

This is different from the `Logout Reminder Page` feature which also show a page when the user comes back after closing the browser.

The page contains a heading, an info message and a continue button. You can customize them via `Custom Translations` by setting the following labels:

- `title.timeout.page`
- `info.timeout.page`
- `continue.button.label`

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.

AppleLogin_scope

Select the requested scopes for getting user information from Apple.

The default is `email` and thus minimal information will be returned.

The scope `openid` will always be added as Apple uses OpenID Connect.

RuleBundle_exceptionRules

Configure *exception modifications*.

As explained in the ModSecurity documentation *exception modifications* are applied **after** including the core rules.

If both the `Request Validation Settings` and the `Rule Bundle` pattern have *exception modifications* configured, first the `Request Validation Settings`, then the `Rule Bundle` modifications 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`:

```
SecRuleRemoveById 900200
```

- Whitelist body parameter `upload` for all rules:

```
SecRuleUpdateTargetByTag ".*" "!ARGS:upload"
```

- Whitelist body parameter `upload` for rule ID `123`:

```
SecRuleUpdateTargetById 123 !ARGS:upload
```

- Add a new rule which allows the HTTP methods used for WebDAV:

```
SecAction \
 "id:1,\
  phase:1,\
  nolog,\
  pass,\
  t:none,\
  setvar:'tx.allowed_methods=GET HEAD POST OPTIONS PUT PATCH DELETE CHECKOUT COPY DELETE LOCK MERGE MKACTIVITY MKCOL MOVE PROP
```

FacebookLogin_clientSecret

Client Secret is `App Secret` provided by Facebook when you register Facebook as IdP service.

OAuth2AuthorizationServer_cacheTimeout

Caching of responses from a nevisMeta instance. After this time (in seconds), a response is considered outdated and attempts are made to update it.

GenericDeployment_commandPhase

Defines when the command is executed. The files are always copied during the CONFIGURE phase.

Phases:

- CONFIGURE: Command runs after files have been uploaded, but before NEVIS instances are (re)started. Use e.g. when patching a NEVIS instance configuration file.
- ACTIVATE: Command runs when instances are (re)started. Use when deploying files or commands that are independent of NEVIS instances.

UserInput_label

Enter a text or *litdict key* to be displayed as *label* in front of the input field.

NevisIDMUserLookup_clientId

The source of the client's external ID.

Used only when `Show Client Input Field` is set to `disabled`.

Set either this or `Client Name`.

NevisAuthConnector_languages

Enter the language codes which are enabled in nevisAuth.

This property is considered only if nevisLogrend is generated by nevisAdmin to ensure that nevisLogrend provides support for the same languages.

GenericWebBase_filters

Configure filters and their mappings using the XML syntax described in the nevisProxy Technical Documentation.

Filters that have the same name as other filters (even those defined by other patterns) will be combined: the `init-param` sets will be merged where possible. Direct contradictions are interpreted as validation failures.

Example 1: Create (or patch) a filter with a fixed name

```
<filter>
   <filter-name>SomeName</filter-name>
   <filter-class>ch::nevis::isiweb4::filter::SomeClass</filter-class>
   <init-param>
      <param-name>...</param-name>
      <param-value>...</param-value>
   </init-param>
</filter>
```

Example 2: Create (or patch) a filter using an application-specific name

```
<filter>
   <filter-name>SomeName_${service.name}</filter-name>
   <filter-class>ch::nevis::isiweb4::filter::SomeClass</filter-class>
   ...
</filter>
```

Example 3: Map a filter to a sub-path of the assigned application(s). This example works for applications which have 1 frontend path only.

```
<filter-mapping>
   <filter-name>SomeFilter</filter-name>
   <url-pattern>${service.path}/custom/*</url-pattern>
</filter-mapping>
```

Example 4: Use multi-value expressions

Multi-value expressions replicate an entire line for each associated value.

Use the expressions `*{service.path}` and `*{service.mapping}` to generate filters which must contain the frontend paths of all assigned applications.

The following snippet is not complete but should illustrate the concept:

```
<filter>
    <filter-name>FormSigning</filter-name>
    <filter-class>ch::nevis::isiweb4::filter::validation::EncryptionFilter</filter-class>
    <init-param>
        <param-name>EntryURL</param-name>
        <param-value>
            *{service.path}/
        </param-value>
    </init-param>
    ...
</filter>
```

NevisMetaDeployable_frontendTrustStore

Assign the Trust Store for the HTTPs endpoint.

If no pattern is assigned a Trust Store will be provided by nevisAdmin 4 automatic key management.

CustomNevisMetaLogFile_syslogHost

Defines where to send logs to via syslog.

This configuration is used only when syslog forwarding is enabled (see `Log Targets`).

The syslog facility is `localhost3` and the threshold is `INFO`.

AuthenticationFlow_flow

Assign a step to execute for incoming requests.

If not present already, the step will be added to the `Authentication Realm`.

If no step is assigned the default flow of the `Authentication Realm` will be executed.

NevisAdaptServiceAccessBase_realm

Mandatory setting to enforce authentication.

NevisDetectCoreDeployable_port

Enter the port on which nevisDetect Core will listen.

GenericIngressSettings_caSecretNamespace

Enter the namespace of the `CA Secret`.

GenericAuthRealm_resources

In case your `AuthState` elements require additional configuration files or scripts upload them here.

Files uploaded here will be deployed into the `conf` directory of the nevisAuth instance.

KerberosLogin_kerberosRealms

Enter the allowed `Kerberos realms ( AD domains )`.

Example:

- `SIVEN.CH`

In case multiple values have to be configured you can define which `Keytab File` or `Keytab File Path` to use by referencing its file name.

Example:

- `SIVEN.CH -> kerberos_ch.keytab`
- `SIVEN.DE -> kerberos_de.keytab`

GenericThirdPartyRealm_authServiceAddons

Assign add-on patterns to customize the behaviour of the `Authentication Application` .

Assigning these add-ons here may be more appropriate to have the complete authentication logic concentrated here.

FrontendKerberosLogin_proxyHostNames

Enter the `Frontend Addresses` of the nevisProxy `Virtual Host` patterns for which this pattern provides authentication.

Example:

- `www.siven.ch`

In case multiple values are configured you can define which `Keytab File` or `Keytab File Path` to use by referencing its file name.

Example:

- `www.siven.ch -> kerberos_ch.keytab`
- `www.siven.de -> kerberos_de.keytab`

HostingService_path

The path at which the resources shall be accessible at the frontend. You may use `/` to deploy root content.

LdapLogin_level

Set an authentication level if authentication of this step is successful.

NevisAdaptAnalyzerConfig_browserFingerprintAnalyzer

Used to disable Browser Finger creation and analysis.

StaticContentCache_maxEntries

The maximum number of documents to be cached.

LdapLogin_onSuccess

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

NevisFIDODeployable_frontendTrustStore

Assign the trust store for validating incoming connections on the HTTPS endpoint.

If no pattern is assigned an automatic trust store will be generated. This requires automatic key management to be enabled in the inventory.

In that case clients, such as nevisProxy and nevisAuth, should use an automatic key store for the connection to nevisFIDO so that the trust can be established.

GenericSocialLogin_authorizationEndpoint

The authorization endpoint of the OAuth2 provider.

Required when `Provider Type` is set to `OAuth2`.

GenericThirdPartyRealm_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.

HostContext_keystore

A key store is required when HTTPS is used for any of the `Frontend Addresses`.

If not set a key store will be set up automatically. This requires automatic key management to be enabled.

The key store should provide a certificate which is valid for all frontend addresses, e.g. because SANs (subject alternative names) are set or because it is a wildcard certificate.

In a Kubernetes deployment the TLS connection is terminated by an NGINX Ingress in front of nevisProxy and this configuration does not apply. The required key material will be generated automatically.

You can configure the NGINX Ingress to use key material from a Kubernetes secret as follows:

1. assign a `NGINX Ingress Settings` pattern to your `nevisProxy Instance` via `Additional Settings`
2. in the `NGINX Ingress Settings` pattern configure `TLS Secrets`.

AuthCloudBase_onAbort

Assign a step to continue with when the user has aborted in the mobile app or a timeout occurred.

CustomAuthLogFile_auditChannels

Assign `nevisAuth Audit Channel` patterns to use your own channel implementations.

FIDO2Onboarding_onCancel

If assigned a skip button will be added.

Use to provide an alternative to the user.

The button is defined by the label `info.signup.passwordless.skip` and looks like a link.

Translations for this label must include a button with name `cancel-bottom`. Example:

```
<button name="cancel-bottom" type="submit" value="true" class="btn btn-link link-primary">Skip for now</button>
```

OutOfBandMobileDeviceRegistration_realm

Assign an `Authentication Realm` to protect the APIs for out-of-band registration.

When the APIs are called by a protected application which is exposed / running on nevisProxy, then you should assign the same realm here.

AuthCloudBase_accessKey

Instead of uploading an `access-key.json` , you can enter the access key of your Authentication Cloud instance here.

CustomNevisIDMLogFile_levels

Configure log levels.

See the nevisIDM Technical Documentation, chapter nevisIDM log levels (file: logging.yml) for details.

Hint: If you only change log levels nevisAdmin 4 does not restart the component in classic VM deployment. The new log configuration will be reloaded within 60 seconds after deployment.

The default configuration is:

```
ch.nevis.idm.batch.jobs = INFO
ch.nevis.idm.standalone = INFO
```

Examples:

```
ch.adnovum.nevisidm.service.properties = INFO
ch.nevis.ninja = DEBUG
```

NevisIDMUserUpdate_writeEmptyValues

If `enabled`, it is possible to clear user attributes or properties. The value will be overwritten with an empty value.

This is supported only if the corresponding attribute or property is optional.

If `disabled`, empty values are ignored, i.e., the stored value remains unchanged.

NevisIDMURLTicketConsume_onSuccess

Assign an authentication step which shall be executed when the URL ticket is valid.

Note: this pattern does not provide any content on the exposed `Frontend Path(s)` and does not ensure that the caller is redirected when the authentication flow terminates.

Thus, please take appropriate measures at the end of the flow to avoid a `404` error. For instance, you may trigger a redirect at the end of your flow, or assign an `URL Handler` to `Additional Settings`.

StaticContentCache_maxAgeMode

Choose one of:

- **override**: The cache entry lifetime is set to **Max Lifetime**.
- **backend**: The cache entry lifetime is copied from the `Cache-Control: max-age` directive sent by the backend. The **Max Lifetime** is used as a fallback.

SamlSpConnector_type

Reserved for ID Cloud use.

NevisIDMUserLookup_buttons

Assign a `Dispatcher Button` to add button(s) which points to a different authentication step.

OAuth2AuthorizationServer_idTokenJWKSetTrust

Assign a trust store for the outbound TLS connection to JWK Set endpoint for ID Token encryption.

Import the CA certificate of the `JWK Set endpoint` 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.

AuthorizationPolicy_requiredRolesMode

The `Required Roles Mode` defines which `Required Roles` are set for the current paths.

When combining several `Authorization Policy` patterns for an application, this setting allow inheriting the `Required Roles` from a more general pattern.

Choose one of:

- `self-contained` : The `Required Roles` defined in this pattern are applied to the current paths. They override any `Required Roles` set on parents paths. If no `Required Roles` are set in the current pattern, no required roles will be enforced for the current paths.
- `inherited` : The `Required Roles` in this pattern is not used. Use this setting if you have another `Authorization Policy` pattern applied to a parent path to inherit the configuration from. For the `Required Roles` to be inherited from a particular parent, this setting has to be set to `default (self-contained)` in the parent pattern (otherwise you may inherit a value from a grandparent).

PemKeyStoreProvider_rootDirName

Set to deploy the key store underneath a *base* directory. The key store will be established at:

```
/var/opt/keys/own/<base>/<name>
```

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

NevisIDMUserCreate_mandatoryAttributes

Define which attributes will always be set for the user.

The value can be constant or determined by a nevisAuth or EL expression. User creation will fail when the value is empty.

Which attributes must be provided depends on policy configuration in nevisIDM.

How to best determine the value depends on preceeding authentication states and the (session) variables they produce.

For instance, let's assume that the email is stored in a session variable called `email` , the first name in `firstname` , and the last name in `name` . You can then use:

```
email: ${sess:email}
loginId: ${sess:email}
```

```
  name: ${sess:name}
  firstName: ${sess:firstname}
```

InBandMobileAuthenticationRealm_tokens

Tokens assigned here may be created after successful authentication.

To produce and forward a token to an application backend, reference the same token from the application's `Additional Settings` property.

SAPLogonTicket_recipientSID

See SAP documentation of SAP SSO logon tickets for more information. Setting no value for this property should be correct for most cases.

NevisDPDeployable_idmKeystore

Assign a key store which shall be used for outbound (2-way) TLS connections to nevisIDM. If no pattern is assigned no key store will be generated.

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

```
  ${idm.keystore}
  ${idm.keystore.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.keyStore" value="${idm.keystore}"/>
          <value name="javax.net.ssl.keyStorePassword" value="${idm.keystore.password}"/>
          ...
      </dp:paraMap>
  </object>
```

DeployableBase_memoryLimit

This setting defines the maximum amount of RAM than can be used by this instance.

VM Deployment

By default, the Java process will use 1/4 of the available RAM.

Depending on how many instances are deployed to the same target host this may be either **too much** or **too little**.

The value configured here will be used for the maximum heap size of the Java process (`-Xmx`).

Kubernetes Deployment

In Kubernetes deployment the value configured here will be ignored and the Java process will be configured to use a percentage of the available RAM.

Note that `-Xmx` is not set to avoid file changes when adapting the limit.

As the docker container runs only 1 process the JVM flags `-XX:+UseContainerSupport` and `-XX:MaxRAMPercentage=80.0` will be applied so that Java process can use up to 80% of the configured limit.

JWTToken_attributes

Add custom claims to the JWT token.

Values can be static, nevisAuth expressions (`${...}`) or EL expressions (`#{...}`).

Examples:

Claim	Expression
email	`${sess:user.email}`

FacebookLogin_clientExtId

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

ErrorHandler_contentTypes

The **Content-Types** configures the `Content-Type` headers for which the **Content-Type Mode** setting is applied. Enter one value per line.

Use this setting in combination with the **Content-Type Mode** setting.

NevisLogrendLogSettings_serverLogFormat

Log4j 2 log format for the default SERVER logs.

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

MultipleFieldUserInput_onSuccess

Configure the step to execute after the user has provided input. If no step is configured here the process ends with `AUTH_DONE`.

NevisIDMSecondFactorSelection_oath

Assign a step which may be selected when the user has an OATH (TOTP) credential.

OATH (TOTP) credentials may be used for second factor authentication using an authentication app, e.g. Google Authenticator.

InBandMobileAuthenticationRealm_auth

The `nevisAuth Instance` where the authentication flow will be configured.

PemKeyStoreProvider_keystorePass

Enter a passphrase.

The passphrase will be used to protect sensitive keystore files (`key.pem`, `keystore.pem`, `keystore.jks`, and `keystore.p12`) on the target hosts.

If you do not enter any passphrase a passphrase will be generated.

As the passphrase is considered sensitive information it should not be published with the project. It is therefore required to use a variable and set the value in the inventory (as a secret).

NevisAdaptLogSettings_serverLogFormat

Logback log format for the default SERVER logs. This pattern is used for **non**-kubernetes deployments.

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

MicrosoftLogin_redirectURI

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

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/microsoft/`
- Frontend Address: `https://nevis.net`

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

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.

8.2505.0

Full changelog:

Patterns 8.2505.0 Release Notes - 2025-05-21

TODO

Describe the most important changes here.

Automatic migration

Describe automatic migrations

NevisIDMDeployable_host

Enter a custom host name to listen on.

NevisFIDODeployable_connectionType

Define which APIs nevisFIDO uses when talking to nevisIDM.

Choose between:

- `default` : use the API recommended by Nevis.
- `both` : uses the SOAP API for most use cases and REST for updating login counters.
- `rest` : uses only the REST API.

As of November 2024 our `default` is `both` . The `rest` mode is still experimental, but is expected to become the `default` in the future.

NevisAuthDeployable_startupDelay

Time to wait before checking Kubernetes readiness on startup.

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

Sets `initialDelaySeconds` of the Kubernetes startup probe.

AuthCloudOnboard_onUserExists

Assign an authentication step to continue with when the user exists and has an active authenticator.

If no step is assigned here the authentication flow will fail for such users.

SamlSpConnector_binding

The `Outbound Binding` controls how SAML messages are returned to the service provider.

Choose a binding which is supported by the service provider.

Use `http-redirect` when the SAML `Response` has to be returned using a `302 Redirect`.

Use `http-post` to generate a self-submitting form which produces a `POST` request. This binding is recommended as SAML `Response` messages will include a signature.

When `http-post` is selected, HTML encoding will be applied to the `RelayState` parameter to include it in the self-submitting form. This ensures that the parameter can be returned to the service provider, even when it contains special characters.

GenericDeployment_groupPermission

Read-write permissions for specified group 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`.

OAuth2Scope_label

Enter a label which shall be show when requesting consent for this scope.

You have to provide `Translations` for this label in the `Authentication Realm` associated with the `OAuth 2.0 Authorization Server / OpenID Provider`.

HostContext_encodedSlashes

Choose from:

- `allowed` : URLs containing encoded slashes are allowed and will not be decoded (`AllowEncodedSlashes NoDecode`). Also `URLEncoding` will be set to `false` for each `HttpsConnectorServlet`.
- `forbidden` : URLs containing encoded slashes will be denied and a 404 will be returned. This is the default behaviour of Apache.

OutOfBandMobileStepBase_clientKeyStore

Assign a key store for the TLS connection to nevisFIDO.

If no pattern is assigned, a key store will be provided by automatic key management.

The client certificate in the key store must be trusted by nevisFIDO.

In case both sides use automatic key management, trust can be established automatically and there is nothing to configure.

However, if you are using a different kind of key store, then you **must** configure `Frontend Trust Store` in the associated `nevisFIDO UAF Instance`.

NevisAdaptAuthenticationConnectorStep_onLogoutDone

Optional. Reference for the next step in the logout authentication flow. If missing, this is the last step and the result will be `AUTH_DONE`.

LdapLogin_userFilter

Enter an LDAP Filter.

Use when the user has to be determined with custom criteria. When configured this is used instead of `User Attribute`.

Example:

```
(|(${notes:userid}=cn)(${notes:userid}=mail))
```

For debugging the authentication set the log level of `JNDI` to `DEBUG`.

NevisIDMTermsAcceptance_nevisIDM

Reference a `nevisIDM Instance` to be used for checking terms and conditions.

NevisIDMDeployable_frontendKeyStore

Assign the Key Store provider for the HTTPs endpoint. If no pattern is assigned a Key Store will be provided by the nevisAdmin 4 PKI.

NevisFIDODeployable_serverTrustStore

Assign the trust store for validating the nevisIDM endpoint.

The trust store should contain the certificate of the CA that has issued the server certificate.

If no pattern is assigned an automatic trust store will be generated. This requires automatic key management to be enabled in the inventory.

In that case the pattern assigned to `nevisIDM` must be a `nevisIDM Instance` pattern which uses an automatic key store for the `Frontend Key Store`.

TransformVariablesStep_properties

Set property elements for the `TransformAttributes` state.

SwissPhoneChannel_defaultCountryCode

The default country code to add to the mobile number if the number found in the session does not have a country code.

This value must **not** contain a `+`.

For instance, assuming that numbers without country code information are Swiss, enter `0041` in this field.

NevisDetectLogSettings_levels

Configure log levels.

See nevisDetect Reference Guide, chapter `Logging Configuration` for details.

Hint: If you only change log levels nevisAdmin 4 does not restart the component in classic VM deployment. The new log configuration will be reloaded within 60 seconds after deployment.

The default configuration is:

```
ch.nevis.nevisadapt = INFO
ch.nevis.nevisdetect.util.logging.OpTracer = DEBUG
```

Examples:

```
org.springframework.web.filter.CommonsRequestLoggingFilter=DEBUG
ch.nevis.nevisdetect.entrypoint.icap.RequestProcessingHelper=INFO
```

MicrosoftLogin_type

The application type that you choose when you create your application in Microsoft. We are supporting 3 types

- common
- organizations
- consumers

Please follow this document to select your application type correctly

LdapLogin_onFailure

Assign an authentication step that is processed if LDAP authentication fails with an technical error, or if the user is not unique.

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

EmailInputField_label

Enter a text or *litdict key* to be displayed as *label* in front of the input field.

OutOfBandManagementApp_addons

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

NevisIDMPasswordLogin_loginType

Define how to look up the user.

Choose between:

- `LOGINID` - lookup user by `loginId` attribute.
- `EMAIL` - lookup user by `email` attribute.
- `AUTO` - depending on what has been entered, nevisIDM tries to look up the user by `email` or `loginId` attribute.

We recommend to use `LOGINID` as it is the most efficient way to look up users and has no side effects. This can even work when users enter their email as you can store the email in the `loginId` attribute as well.

For `AUTO` and `EMAIL` to work nevisIDM has to be configured accordingly. You either have to:

- Set `authentication.loginWithEmail.enabled=true` in the Client policy. Policies cannot be configured using patterns. You can change them on the nevisIDM Admin GUI.
- Set `application.feature.emaillogin.enabled=true` in `nevisidm-prod.properties`. Use the `Generic nevisIDM Instance Settings` pattern for this.

NevisIDMPasswordLogin_onSuccess

Configure the step to execute after successful authentication.

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

4.18.0

Full changelog:

Patterns 4.18.0 Release Notes - 2023-02-15

Mobile Authentication Pattern Improvements

The mobile authentication patterns have been improved. The help texts have been completely rewritten to give more details about the use case. New patterns have been added to make the solution more flexible. The existing `Mobile Device Registration` pattern has been marked as deprecated and will be removed in the May 23 release. If you are using mobile authentication, see release notes for more details.

New Database Patterns

Introducing new database patterns for all components which use a database.

The new patterns can be used in classic and Kubernetes setups.

Depending on the deployment type, different settings will be used.

Migration to the new patterns is **mandatory**.

There will be error issues guiding you through the process.

nevisAuth Session Store Changes

There are breaking changes in the local and remote session store configuration in nevisAuth 4.38. See nevisAuth release notes for details. You must upgrade nevisAuth to 4.38 when using patterns 4.18 and vice versa.

nevisAuth Outbound HTTP Changes

There are breaking changes in the in nevisAuth 4.38 affecting the behaviour of outbound HTTP connections. See nevisAuth release notes for details. You must upgrade nevisAuth to 4.38 when using patterns 4.18 and vice versa.

NevisIDMUserLookup_clientName

The source of the client's name.

Used only when `Show Client Input Field` is set to `disabled`.

Set either this or `Client ID`. When neither is set then `Default` is used.

NevisDetectLogSettings_serverSyslogFormat

Logback log format for the SERVER SYS logs.

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

AuthCloudBase_skipLabel

Label to display on the element which allows the user to skip.

The element is usually a button but this can be changed by setting `Skip Type`.

NevisAdaptAuthenticationConnectorStep_adapt

Reference for the nevisAdapt service to calculate risk scores during authentication.

Logout_logoutBehaviour

- `gui` - shows a logout GUI. On submit the user is redirected to the same URL with the query parameter `logout` removed.

- `redirect` - does not show a GUI. The user is immediately redirected to the given URL or path.

AutomaticTrustStoreProvider_truststoreFile

Upload additional trusted certificates in PEM format.

The content of all files will be concatenated and added to the `truststore.*` files generated by this pattern.

You can make this a variable and upload the files in the inventory using the `Attach files` function.

SamlSpIntegration_token

Assign a SAML Token pattern.

The referred pattern must:

- have `Token Type` set to `Response`
- be assigned to the correct Realm pattern(s)

AuthCloudLogin_onSuccess

Assign a step to execute after successful authentication.

If no step is configured, the flow ends and an authenticated session will be established.

This requires that the session contains an authenticated user.

A simple way to ensure that is to include `nevisIDM User Lookup` or `nevisIDM Password Login` steps in your flow.

LdapLogin_onUserNotFound

Assign an authentication step to be invoked if the user could not be found.

For instance, you may use this setting to chain multiple `LDAP Login` patterns, e.g. to lookup users based on a different `User Attribute` or in separate `LDAP Endpoints`.

The following `notes` will also be set and may be shown if the next state renders a GUI:

```
lasterror               = 1
lasterrorinfo           = authentication failed, invalid input
lastresult              = usernotfound
```

NevisFIDODatabase_schemaPassword

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

WebhookCalls_authMode

Configure the `authMode` property.

InBandMobileAuthenticationRealm_fidoTrustStore

Assign a pattern which provides the trust store for nevisAuth to connect to nevisFIDO.

HeaderCustomization_basicAuthUser

Enter the basic auth user or an expression of the format `<source>:<parameter>`.

For the `<source>` you may use:

- `AUTH`: outargs returned by nevisAuth.
- `CONST`: constant strings.
- `ENV`: Apache environment variables.
- `PARAM`: values from a request body as provided by a `ParameterFilter`.
- `HEADER`: request headers.

NevisAdaptPluginPattern_propagateDeviceRecognition

Risk scores to be delivered to the client in the request headers. This option configures enables device cookie risk score to be propagated.

MicrosoftLogin_tenantId

Enter the `Tenant ID` of your Azure Active Directory.

This setting is used when `Application Type` is set to `organizations`.

Check Microsoft documentation on How to find your Azure Active Directory tenant ID.

JWTToken_properties

Set low-level properties for the JWTToken AuthState.

SOAPServiceAccess_schemaValidation

Choose between:

- `strict` - **all** requests must be valid SOAP, requests without a body are blocked.
- `enabled` - **all** requests which have a request body must be valid SOAP, requests without a body are allowed.
- `log only` - similar to `strict` but violations are not blocked, only logged.
- `content-type` - validation is performed only when the `Content-Type` header matches `application/soap+xml`.
- `disabled`

ErrorHandler_keepSecurityHeaders

Configure the name of special response headers which should be kept, regardless of the header action of the matching rule. Useful for keeping the security response headers for the error pages.

Default:

```
Strict-Transport-Security
X-Content-Type-Options
Referrer-Policy
```

GroovyScriptStep_errorStatusCode

Set the status code for responses when the `Response Type` is set to `AUTH_ERROR`.

The default of `403` is backward compatible.

Note that we generally use `403` for unhandled error cases in authentication step patterns. This is to avoid exposing the information that a certain case is not properly handled.

Depending on your case, a `500` or `400` may be a more appropriate choice.

SamlResponseConsumer_issuer

Configure the `Issuer` used by this SAML Service Provider (SP).

This setting is used only when Artifact Binding is used.

Example: `https://sp.example.org/SAML2`

GenericDeployment_templateFiles

Expressions matching files in which to replace parameters.

If a single .zip file is unpacked, it is scanned for matching files as well.

Possible values are exact file names or file endings.

Example:

- my_script.sh
- *.txt
- *.properties

TokenHeaderPropagation_header

Enter the HTTP header to set for requests to backend applications. The value of this header will be the base64 encoded token.

NevisAuthRealmBase_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.

NevisProxyDeployable_addons

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

NevisDetectPersistencyWebApplicationAccess_persistency

Reference for the pattern with the details of the web application.

Supported patterns:

- nevisDetect Persistency Instance

ICAPScanning_subPaths

Set to apply the ICAP scanning on some sub-paths only.

Sub-paths must be relative (e.g. not starting with `/`) and will be appended to the frontend path(s) of the virtual host (`/`) or applications this pattern is assigned to.

Sub-paths ending with `/` are treated as a prefix, otherwise an exact filter-mapping will be created.

The following table provides examples to illustrate the behaviour:

Frontend Path	Sub-Path	Effective Filter Mapping
`/`	`secure/`	`/secure/*`
`/`	`accounts`	`/accounts`
`/`	`api/secure/`	`/api/secure/*`
`/`	`api/accounts`	`/api/accounts`
`/app/`	`secure/`	`/app/secure/*`
`/app/`	`accounts`	`/app/accounts`
`/app/`	`api/secure/`	`/app/api/secure/*`
`/app/`	`api/accounts`	`/app/api/accounts`

NevisIDMUserUpdate_optionalAttributes

Define which attributes are optional and how to provide them.

Example:

```
firstName: ${sess:given_name}
name: ${sess:family_name}
country: ${sess:country}
```

NevisProxyDatabase_accessRestriction

This optional configuration is available when `Mode` is set to `hybrid`.

Assign an `Access Restriction` pattern to define the source IPs that are allowed to access the `Session Store Path`.

7.2311.0

Full changelog:

Patterns 7.2311.0 Release Notes - 2023-11-15

Kubernetes Deployments

We have added the `Crash Recovery Strategy` `kill` in the `nevisProxy Instance` pattern. For Kubernetes deployments this is the new default.

SAML Signing and Signature Validation

We have refactored the `Signature Validation` in `SAML IDP Connector` and `Signed Element` in `SAML SP Connector` to provide more options. The new drop-downs are multi-select and thus you can choose multiple options.

The option `both` has been removed as more than 2 signed elements can be configured. The default in the `SAML IDP Connector` has changed.

OAuth2AuthorizationServer_authCodeLifetime

How long an authorization code issued by the authorization server should be valid.

DeployableBase_startInactive

In a classic VM deployment the instance is restarted when a configuration file changes that requires a restart. The instance is not restarted when a configuration file changes that does not require a restart.

This setting defines if the instance should also be started when it is down.

This setting applies to classic VM deployment only. In Kubernetes deployment the container pods are always recreated when any configuration file changes.

NevisIDMDatabase_jdbcDriver

Due to licensing restrictions, we cannot ship any Oracle dependencies. If you want to use an Oracle database, upload a JDBC driver here.

The driver can be downloaded from Oracle.

Note that both the component (`nevisidm`) and the database migration tool (`nevisidmdb`) need a JDBC driver to access the database. In a classic deployment, the driver will therefore be added to 2 different instance directories.

In Kubernetes setups, and when `Database Management` is enabled, you have to configure `Volume Claim` **instead** of uploading the JDBC driver here. This is to avoid committing binary files to Git during the deployment process.

NevisAdaptFeedbackConfig_proxy

Reference for the nevisProxy instance to set up frontend addresses.

SocialLoginBase_nevisIDM

Choose which nevisIDM instance you want to store the user's information after logged in with social login provider.

NevisLogrendLogSettings_serverSyslogFormat

Log4j 2 log format for the SERVER SYS logs.

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

NevisIDMPruneHistoryJob_cronExpression

Enter a cron expression which defines when this job will be executed.

Cron expressions consist of 6 required fields and one optional field separated by white space.

The field order is:

1. Seconds
2. Minutes
3. Hours
4. Day-of-Month
5. Month
6. Day-of-Week
7. Year (optional)

Cron expression can be complex and this pattern only validates the length. The most important wildcards are:

- `*` is used to specify all values. For example, `*` in the minute field means *every minute*.
- `?` is allowed for the day-of-month and day-of-week fields. It is used to specify *no specific value*.
- `-` is used to specify ranges.

Further information about the supported syntax can be found in the javadoc of org.quartz.CronExpression.

Examples:

- `0 0 0 * * ?` : fires every midnight.
- `0 0/30 8-9 5,20 * ?` : fires every half hour between the hours of 8 am and 10 am on the 5th and 20th of every month.
- `0 30 10-13 ? * WED,FRI` : fires at 10:30, 11:30, 12:30, and 13:30, on every Wednesday and Friday.

SecurosysKeyStoreProvider_securosysPin

The PIN for accessing the materials on the HSM.

You must set it as a variable for security reasons.

TCPSettings_keepAliveByClient

Forces TCP connections to only be reused for the same client. A call from a different client will use another TCP connection from the connection pool. If set to `default` , the nevisProxy default will be used.

OAuth2UserInfo_path

Enter the path where the endpoint shall be exposed on nevisProxy.

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.

OutOfBandManagementApp_realm

Configure an authentication realm, which will protect the device management application.

GenericSocialLogin_providerType

The provider type of the social account: either `OpenID Connect` or `OAuth2`.

AuthServiceBase_host

Assign a `Virtual Host`.

NevisAdaptDeployable_deviceCookieName

Provide a name for the cookie that will be used as the volatile identification for a browser.

Leave this configuration empty if you want to keep the default value of `DEVICE_COOKIE`.

LogSettingsBase_rotationType

Select log rotation type.

Choose between:

- `size` - defines the maximum file size before the log files are rolled over
- `time` - defines the time span after which logs are rolled over

If you rotate by time we recommend you monitor the disk usage as log files can be huge.

Note: a combination of size and time based log rotation is not supported.

OnDemandEntry_level

Define the authentication level that this flow produces on successful execution.

The step assigned to `On Entry` (or a subsequent step) must achieve at least this level.

NevisAdaptDeployable_feedbackConfig

Provide additional settings for defining the details of the distrust session mechanism:

- JWE key to generate new tokens with
- nevisAuth reference to distrust and terminate sessions there as well
- nevisProxy reference to build the distrust feedback URI
- action to take on received token
- token lifetime
- redirect URL after sending the token

OAuth2Client_clientId

Enter the Client ID.

HostContext_listen

The physical address(es) to bind on, with scheme HTTP or HTTPS and ports.

Must be set when multiple virtual hosts should listen on the same endpoint (name-based virtual hosts).

If not set the `Frontend Addresses` will be used to bind.

The host name must resolve to an IP which is bound to a network interface.

You can also use `0.0.0.0` for the host name to listen on all network interfaces.

Examples:

```
https://www.siven.ch:8443
http://localhost:8080
https://192.168.1.1:443
http://0.0.0.0:80
```

ServiceAccessBase_backends

Enter the complete URLs (scheme, host, port and path) of the backend services.

Note:

- all URLs must use the same scheme and path.
- automatic path rewriting will be performed when the path differs from the `Frontend Path`.

In case you are setting multiple addresses, the first one will be chosen as the primary resource.

- When the primary resource cannot be accessed, nevisProxy will attempt to use the next resource.
- Even when the primary resource is reachable again, the request will still go to the current resource until the end of the session. However, for new sessions the primary resource is used.

NevisIDMPasswordLogin_reenterExpiredPassword

When the password is expired or has been reset by an administrator, the user is forced to set a new password.

Set this drop-down to `enabled` to force the user to enter the old password again when this happens.

JWTToken_issuer

The issuer (`iss`) is an optional claim which may be checked by applications receiving this token.

GenericSMTPChannel_smtpUser

If a username is required at the SMTP server enter it here.

NevisIDMUserUpdate_onSuccess

Define how to continue after user update.

RoleCheck_found

Assign a step to continue with when the user has **any** of the configured roles.

If no step is assigned, the authentication flow will be done and the user is authenticated.

DatabaseBase_rootCredentialNamespace

Set if the `Root Credential` is in a different Kubernetes namespace.

OAuth2AuthorizationServer_tokenEndpoint

The endpoint to exchange the authorization code for 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.

NevisIDMServiceAccessBase_backendTrustStore

Assign a trust store if you want to validate the server certificate used by nevisIDM. If this not set, the connection is 1-way TLS.

OutOfBandManagementApp_path

The path at which the management app shall be accessible at the frontend.

NevisDetectCoreDeployable_persistency

Add reference for a nevisDetect Persistency Instance pattern.

NevisIDMDeployable_smtpTLSMode

Choose between:

- `disabled` - SSL/TLS is disabled. The `SMTP Trust Store` is not used.
- `STARTTLS` - uses the `STARTTLS` command (see RFC 2487) to switch to SSL/TLS if supported by the SMTP server.

NevisIDMServiceAccessBase_nevisIDM

References a nevisIDM Instance.

OAuth2AuthorizationServer_invalidRedirectUri

Configure the step to execute when the `redirect_uri` request parameter value is not registered for the client sending the request.

If no step is configured here the flow ends and an error will be displayed.

NevisAuthDatabase_passwordType

Choose between:

- `automatic` : behaves like `command` when the password starts with `/opt/` .
- `command` : use when the input is a command. In `esauth4.xml` the prefix `pipe://` will be added.
- `plain` : take the password as-is.

SamlIdpConnector_audienceCheck

Define how to validate the optional `Audience` element of received SAML assertions.

- `disabled` - `Audience` is not checked
- `lax` - if present the `Audience` has to match the `Allowed Audience`
- `strict` - the `Audience` element must be present and must match the `Allowed Audience`

SharedStorageSettings_storageClassName

The name of the StorageClass. The selected storage should support ReadWriteMany access.

For example: `azurefile`

For more information regarding persistent volume types in Kubernetes please visit this page

HostingService_resources

Upload your resources here.

All files will be deployed in the same directory. Please use standard extensions (e.g. .css, .png, .html, .htm) only.

If you want to use subdirectories please upload a .zip file instead. The content of the .zip file will be unpacked.

KeyObject_properties

Add `property` child elements to the `KeyObject` element.

NevisFIDODeployable_logging

Assign a pattern to customize the log configuration.

LdapLogin_delegateMap

Defines mappings from LDAP attributes to delegate names. The specified LDAP attributes are queried and set as output arguments with the specified output argument name.

- `<attribute-name-in-directory>:<output-argument-name>`
- `<attribute-name-in-directory>`

Examples:

- `givenName`
- `mail:email`
- `telephoneNumber:user.mobile`

NevisDetectRiskPluginBase_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.

NevisIDMChangePassword_nowLocked

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

NevisAdaptDeployable_ipReputationUpload

Provide a file attachment for the IP reputation service to use.

Please consider uploading the file manually if its size exceeds 20MB, then adjust the path `ipReputationMappingFile` in *nevisadapt.properties* after deployment if needed.

Every line should contain a single blacklisted IPv4 range in CIDR format:

`A.B.C.D/E` or `A.B.C.D` (A/B/C/D: [0-255]; E: [0-32])

The IP ranges should not intersect each other.

The IP-mapping file has to be updated regularly for the service to stay relevant. We recommend setting up periodic update of IP geolocation and reputation mappings.

NevisIDMAuthorizationsAddon_rolePermissionsFile

Add properties for `rolesMapping.properties` . If a role not defined in the uploaded file default values will be used for it.

See Functional authorization - nevisIDM roles for details.

The following permissions are allowed:

- ApplicationCreate, ApplicationDelete, ApplicationModify, ApplicationSearch, ApplicationView
- AuthorizationApplCreate, AuthorizationApplDelete, AuthorizationApplSearch, AuthorizationApplView,
- AuthorizationClientCreate, AuthorizationClientDelete, AuthorizationClientSearch, AuthorizationClientView
- AuthorizationCreate, AuthorizationDelete, AuthorizationModify, AuthorizationSearch
- AuthorizationEnterpriseRoleCreate, AuthorizationEnterpriseRoleDelete, AuthorizationEnterpriseRoleSearch, AuthorizationEnterpriseRoleView
- AuthorizationUnitCreate, AuthorizationUnitDelete, AuthorizationUnitSearch, AuthorizationUnitView
- AuthorizationView,
- BatchJobExecute, BatchJobView
- ClientApplAssign, ClientApplDelete, ClientApplView
- ClientCreate, ClientDelete, ClientModify, ClientSearch, ClientView
- CollectionCreate, CollectionDelete, CollectionModify, CollectionView
- ConsentView

- CredentialChangeState, CredentialCreate, CredentialDelete, CredentialModify, CredentialPdfView, CredentialSearch, CredentialView, CredentialViewPlainValue
- EnterpriseAuthorizationCreate, EnterpriseAuthorizationDelete, EnterpriseAuthorizationModify, EnterpriseAuthorizationSearch, EnterpriseAuthorizationView
- EnterpriseRoleCreate, EnterpriseRoleDelete, EnterpriseRoleModify, EnterpriseRoleSearch, EnterpriseRoleView
- EnterpriseRoleMemberCreate, EnterpriseRoleMemberDelete, EnterpriseRoleMemberSearch
- EntityAttributeAccessOverride
- GenerateReport
- HistoryView
- LoginIdOverride, LoginIdModify
- PersistentQueueView, PersistentQueueDelete
- PersonalQuestionCreate, PersonalQuestionDelete, PersonalQuestionModify, PersonalQuestionView, PersonalQuestionSearch
- PolicyConfigurationCreate, PolicyConfigurationDelete, PolicyConfigurationModify, PolicyConfigurationSearch, PolicyConfigurationView
- ProfileArchive, ProfileCreate, ProfileDelete, ProfileModify, ProfileSearch, ProfileView
- DeputyCreate, DeputyDelete
- PropertyAllowedValueCreate, PropertyAllowedValueDelete, PropertyAllowedValueModify, PropertyAllowedValueSearch, PropertyAllowedValueView, PropertyAttributeAccessOverride
- PropertyView, PropertyCreate, PropertyDelete, PropertyModify, PropertySearch
- PropertyValueCreate, PropertyValueDelete, PropertyValueModify, PropertyValueSearch, PropertyValueView
- RoleCreate, RoleDelete, RoleModify, RoleSearch, RoleView
- SearchResultsExport
- SelfAdmin
- TemplateView, TemplateCreate, TemplateDelete, TemplateModify, TemplateStore
- TemplateTextCreate, TemplateTextDelete, TemplateTextModify, TemplateTextView
- TermsCreate, TermsDelete, TermsModify, TermsView
- UnitCreate, UnitCreateTopUnit, UnitDelete, UnitModify, UnitSearch, UnitView

- UnitCredPolicyCreate, UnitCredPolicyDelete, UnitCredPolicyView
- UserArchive, UserCreate, UserDelete, UserModify, UserSearch, UserView
- UserCreateTechUser, UserModifyTechUser, UserDeleteTechUser, UserArchiveTechUser

User related fine-grained permissions

For permissions `UserModify` and `UserView` a more fine-grained permission can be used.

See Configuration of fine-grained permissions for details.

Credential-type specific permissions

For permissions related to credentials (such as CredentialChangeState, CredentialCreate, CredentialDelete, CredentialModify, CredentialPdfView, CredentialSearch, CredentialView, and CredentialViewPlainValue), it's permissible to reduce the elementary permission to specific credential type(s).

See Credential-type specific permissions of nevisIDM roles for details

BehavioSecPluginPattern_flagDescMappings

List of BehavioSec report flag names with their description name in the following format: `<flagName>=<descriptionName>` . Please add each entry line-by-line.

If any of these flags contain true value in the report, it will be added to the respective header field along with the mapped description value.

To delete a default mapping, omit the description field's name: `<flagName>=` . If the flag is part of the default mapping, it will be overwritten, otherwise added.

Default combined values (flag name/description name):

- advancedUser/advancedUserScore

- deviceChanged/deviceDesc
- deviceIntegrity/deviceIntegrityDesc
- diError/diDesc
- finalized/finalizeTimestamp
- isBot/botDesc
- isDuplicate/duplicateDesc
- isRemoteAccess/raDesc
- isReplay/replayDesc
- isSessionCorrupted/isSessionCorruptedDesc
- locationMismatch/locationMismatchDesc
- newCountry/ipCountry
- numpadUsed/numpadRatio
- otjsError/otjsDesc
- pdError/pdDesc
- pocUsed/pocRatio
- tabUsed/tabRatio
- travelTooFast/travelTooFastDesc
- uiConfidenceFlag/uiConfidence
- uiScoreFlag/uiScore

UserInformation_title

Enter a label for the title. No expressions are supported.

By default, the label `title.login` is used. This label is which is translated as `Login` in all languages.

We recommend to use a different label depending on your use case.

Translations for the label can be defined in the realm pattern.

GoogleLogin_claimsRequest

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

DummyLogin_level

Set an authentication level.

NevisFIDODeployable_frontendKeyStore

Assign the key store to be used for the HTTPS endpoint.

If no pattern is assigned a key store will be generated. This requires automatic key management to be enabled in the inventory.

NevisProxyDeployable_crashRecoveryStrategy

Defines how to handle process crashes.

Choose between:

- `recommended` : uses `recover` for classic and `kill` for Kubernetes deployments;
- `recover` : the child process is recovered by the parent process;
- `block` : every request will be blocked by `503 Service Unavailable` status code;

- `kill` : the whole nevisproxy process (including the parent process) is killed. This works only if the owner of the child process has the permissions to kill the parent process (for example in some Kubernetes setups). Otherwise, this option works like `block` .

Note for `block` and `kill` : these actions take place if at least one request was being processed when the crash occurred. These features can be useful for liveness test in Kubernetes setups, so the given pod can be terminated normally in case of a crash. Using one of these options in a classic setup requires to restart nevisProxy with an external tool after a crash.

GenericAuthenticationStep_level

Optionally define an authentication level which will be set if the user has passed this step successfully.

GoogleLogin_scope

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

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

Scope `offline_access` for generate refresh token. This scope will transfer to `access_type=offline` request parameter for matching with Google spec

HostContext_truststore

Set a trust store to validate client certificates for incoming TLS connections.

The trust store may contain an arbitrary number of CA certificates. Client certificates must be signed by those CAs.

Caution: client certificate authentication is not enabled automatically. As of release 4.3.1 there are no dedicated patterns but client cert authentication can be enforced for the entire host (e.g. using `Generic nevisProxy Settings`) or in the authentication process (`X509State` and `IdentityCreationFilter` init-param `ClientCert`).

JWTToken_subject

Enter a nevisAuth expression for the claim `sub`. The default refers to the ID of the authenticated user.

NevisIDMUpdateUserLoginInfo_onFailure

Assign a step to execute if the nevisIDM is not able to update a User's login info.

For instance, you may assign the following steps:

- `User Information`: show an error message and terminate the authentication flow.

GenericSocialLogin_clientSecretMethod

The method used for authenticating the client. It can be either `Basic Authentication` or `POST`. The default value is `Basic Authentication`.

NevisIDMServiceAccessBase_backendKeyStore

Assign a key store if you want to use 2-way TLS for the connection between nevisProxy and nevisIDM.

SecretTestAddon_secretFiles

Set a variable and upload secret file(s) in the inventory.

The file `/var/opt/nevisproxy/<instanceName>/run/secret_files.txt` should then contain:

- classic: resolved value(s)

- Kubernetes: `inv-res-secret://` reference(s)

OAuth2Client_secret

Enter the Client Secret.

Configuration is required for certain flows only.

GenericDeployment_ownerPermission

Read-write permissions for specified owner 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`.

GenericHostContextSettings_removeFilterMappings

Remove `<filter-mapping>` elements generated by other patterns.

This is an advanced configuration. Use only when you want to remove a `<filter-mapping>` but keep the `<filter>` element, e.g. to map it on a sub-location.

The syntax is a map of `<filter-name>:<url-pattern>`, according to values from the `web.xml`.

For instance, the following would remove the `ErrorHandler_Default` from `/*`:

```
ErrorHandler_Default:/*
```

OAuth2AuthorizationServer_oidc

If enabled the scope openid is allowed for this client.

NevisAdaptDeployable_suspiciousCountryCodeList

Provide a list of two-letter ISO country codes of considerable risk.

Input method 1: Single line - comma-delimited

Input method 2: One country code entry per line

ISO code description can be found at: https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

GenericSMTPChannel_smtp

Enter host:port of the SMTP server.

NevisIDMPasswordLogin_separateScreens

Set to `enabled` to ask for the username and password in two separate screens.

InBandMobileAuthenticationRealm_fidoKeyStore

Assign a pattern which provides the key store for nevisAuth to connect to nevisFIDO with client TLS.

OATHOnboarding_onSuccess

Assign a step to execute **after** onboarding the authenticator app.

We recommended to assign `OATH Authentication` to validate that the onboarding was successful.

Also note that the `nevisIDM Second-Factor Selection` pattern only considers OATH credentials which have passed the `OATH Authentication` once.

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

NevisIDMDatabase_oracleVolumeClaimName

Due to licensing restrictions, we cannot ship any Oracle dependencies.

If you are using an Oracle database, are deploying to Kubernetes, and `Database Management` is *enabled* (`complete` or `schema`), then you have to provide a Kubernetes volume containing an Oracle driver and client.

For more information, see Preparing Oracle Volume.

Enter the name of that volume here.

The volume will be mounted in the `nevisidm-dbschema` image to set up and patch the database schema.

The volume will be mounted in the `nevisidm` image to connect to the database. Because of that, there is no need to upload a `JDBC Driver` .

Maintenance_start

Enter the start date and time of the maintenance window.

- format: `yyyy-mm-dd HH:mm` (24 hours)
- timezone: UTC (not your local time)
- example: `2020-05-20 15:00`

OAuth2Client_profile

Set to `allowed` to allow this client to request the scope `profile`.

This scope produces various claims.

OATHOnboarding_nevisIDM

Reference the nevisIDM Instance which has been used for first factor authentication.

HeaderCustomization_requestPhase

- `BEFORE_SANITIATION` - manipulate request headers early to hide them from validation and authentication.
- `AFTER_AUTHENTICATION` - the original values are subject to validation and can be accessed in the authentication flow. The header manipulation is applied afterwards to affect the application only.
- `END` - manipulate request headers late, just before the request is forwarded to the application.

NevisIDMProperty_propertyScope

Select the type of property:

- `USER_GLOBAL` : all users have this property
- `CREDENTIAL_GENERIC_GLOBAL` : all `Generic` credentials have this property
- `UNIT_GLOBAL` : all units have this property

GenericWebBase_parameters

Define *Template Parameters*.

Examples:

```
backend-host: backend.siven.ch
```

These parameters can be used in:

- `Servlets and Mappings`
- `Filters and Mappings`

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 `\}` .

TestingService_onPlanning

Use for testing only.

NevisAdaptRememberMeConnectorStep_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.

NevisAuthRealmBase_sessionTracking

Choose between:

- `COOKIE` : issue a session cookie.
- `AUTHORIZATION_HEADER` : track the session based on the value of the Authorization header.
- `CUSTOM` : track the session based on custom configuration. It generates an empty session filter which has to be replaced (see below).
- `disabled` : disable session tracking.

CUSTOM session tracking

Given a pattern name of SSO, the following empty filter will be generated:

```
<filter>
    <filter-name>SessionHandler_SSO</filter-name>
    <filter-class>__REPLACE_USING_GENERIC__</filter-class>
</filter>
```

For the filter-class, a placeholder (**REPLACE_USING_GENERIC**) will be used and that placeholder has to be overwritten.

Another pattern must complete the session filter. For example, use `Generic Virtual Host Context` pattern with the following Filters and Mappings configuration:

```
<filter>
    <filter-name>SessionHandler_SSO_RealmName</filter-name>
    <filter-class>ch::nevis::nevisproxy::filter::session::SessionManagementFilter</filter-class>
    <init-param>
        <param-name>Identification</param-name>
        <param-value>CUSTOM</param-value>
    </init-param>
    <init-param>
        <param-name>Custom.RequiredIdentifiers</param-name>
        <param-value>HEADER:Authorization</param-value>
    </init-param>
    <init-param>
        <param-name>Servlet</param-name>
        <param-value>LocalSessionStoreServlet</param-value>
    </init-param>
</filter>
```

RequestValidationSettings_ruleBundle

Add a Rule Bundle pattern for global or group ModSecurity rule configuration.

NevisLogrendDeployable_logrendProperties

Add or overwrite properties in `logrend.properties`.

You can use the following expressions (format: `${...}`):

- `${protocol}` - depends on `HTTPs` setting
- `${host}` - depends on `Bind Host` setting
- `${port}` - will be replaced with `TCP Service Port`

- `${instance}` - contains the instance name

This is an advanced setting which should be used in complex setups only. If you have to configure anything here we are looking forward for your use case.

SocialLoginExtender_socialLogin

Quick win solution for ID Cloud to short-cut the automatic lookup logic.

CustomNevisIDMLogFile_maxFileSize

Maximum allowed file size (in bytes) before rolling over.

Suffixes "KB", "MB" and "GB" are allowed. 10KB = 10240 bytes, etc.

Note: This parameter only applies to `application.log` and `batch.log` (as `audit.log` is configured with `DailyRollingFileAppender`).

NevisDPDeployable_configuration

The `dataporter.xml` must be uploaded here.

Click `Download Configuration Template` to get started.

4.14.0

Full changelog:

Patterns 4.14.0 Release Notes - 2022-02-16)

Log Settings

The configuration options for `Log Targets` have been adapted to fit Kubernetes deployment. If you get an error please select one of the available options.

Support for Syslog forwarding has been deprecated. If you need Syslog forwarding please get in touch to discuss your requirements before May 2022 when Syslog forwarding support will be removed.

The Generic Log Settings patterns have been deprecated. If you are using them please get in touch to discuss your requirements before May 2022 when these patterns will be removed.

nevisIDM Encryption

Since this release the security settings used by NevisIDM to store encrypted properties and URL tickets are exposed. Setting the `Encryption Key` is mandatory to enforce the proper handling of these settings. For setups with existing encrypted data in the database you can enable a fallback mechanism. However, it is recommended to disable the fallback as soon as possible for stronger security.

nevisAuth Client Authentication

Since this release we don't generate `Frontend Trust Store` any more when `Client Authentication` is `disabled` in `nevisAuth Instance` patterns.

This requires nevisAuth version `4.34` or later. You must upgrade nevisAuth before deploying to avoid connection issues.

Realm Translations Encoding

Up to (and including) pattern version 4.13.0 `Translations` in realm patterns supported ASCII encoded files only. By accident ISO-8859-1 encoded files were also working in some scenarios.

Since release 4.13.1 the encoding is fixed to `UTF-8` and thus invalid characters were displayed in some setups.

This release fixes the remaining encoding issues. Only the following encodings are supported:

- ASCII with HTML escaped special characters

- UTF-8

If a character is detected in the uploaded files that cannot be represented in UTF-8 a warning message will be shown.

Further, trying to set `-Dch.nevis.esauth.litdict.charset.encoding` using `Generic nevisAuth Instance Settings` will raise an error.

If you get a warning or error message adapt your pattern configuration accordingly.

AccessRestriction_listingType

Indicates if `Source IPs` should be used as blacklist or whitelist.

- `blacklist` : Access from all configured `Source IPs` is denied. All other IPs are allowed.
- `whitelist` : Access is allowed only for IPs in the `Source IPs` list. All other IPs are blocked.

AuthorizationPolicy_subPaths

Set to apply this pattern on some sub-paths only.

Sub-paths must be relative (e.g. not starting with `/`) and will be appended to the frontend path(s) of the virtual host (`/`) or applications this pattern is assigned to.

Sub-paths ending with `/` are treated as a prefix, otherwise an exact filter-mapping will be created.

The following table provides examples to illustrate the behaviour:

Frontend Path	Sub-Path	Effective Filter Mapping
`/`	`secure/`	`/secure/*`
`/`	`accounts`	`/accounts`

Frontend Path	Sub-Path	Effective Filter Mapping
`/`	`api/secure/`	`/api/secure/*`
`/`	`api/accounts`	`/api/accounts`
`/app/`	`secure/`	`/app/secure/*`
`/app/`	`accounts`	`/app/accounts`
`/app/`	`api/secure/`	`/app/api/secure/*`
`/app/`	`api/accounts`	`/app/api/accounts`

MobileDeviceDeregistration_realm

To provide the best possible security, the nevisFIDO APIs required for mobile device deregistration may be protected by In-Band Authentication.

Assign an `In-band Mobile Authentication Realm` here.

NevisIDMAccountRecovery_nevisIDM

Reference a nevisIDM Instance to be used for checking terms and conditions.

RealmBase_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.

DummyLogin_label

Set to show a different message.

Maintenance_updateInterval

Enter the time interval between checks of the maintenance page.

- In normal mode, the system checks the maintenance page for updates when a request comes in, if the configured interval has passed since the last check.
- In maintenance mode, the system ignores the `UpdateInterval` and fetches the maintenance page on each request.

NevisDPLogSettings_regexFilter

If set, messages for `dataporter.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.*
```

RequestValidationSettings_level

Sets the `paranoia level` of the ModSecurity OWASP Core Rule Set (CRS). Please see https://coreruleset.org/faq/ for more details.

- Paranoia level `1` (PL1) is recommended for beginners and setups with standard security requirements. If you encounter false positives at PL1 OWASP recommends to raise an issue at their Github site.
- Paranoia level `2` (PL2) includes SQL, XSS and code injection rules. PL2 is recommended for setups with elevated security requirements and advanced users.
- Paranoia level `3` (PL3) enables additional rules and keyword lists to cover less common attacks. Consider PL3 if you are experienced at handling false-positives and for sites with high security requirements.
- Paranoia level `4` (PL4) also restricts special characters. PL4 may produce a lot of false positives so please do extensive testing before going into production.

LogSettingsBase_maxFileSize

Maximum allowed file size (in bytes) before rolling over.

Suffixes "KB", "MB" and "GB" are allowed. 10KB = 10240 bytes, etc.

Note: not relevant when rotation type is `time` .

RealmBase_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.

NevisDPLogSettings_levels

Configure log levels.

See nevisDataPorter Technical Documentation, chapter Logging, tracing, debugging, and profiling for details.

Hint: If you only change log levels nevisAdmin 4 does not restart the component in classic VM deployment. The new log configuration will be reloaded within 60 seconds after deployment.

The default configuration is:

```
dataporter = INFO
```

Examples:

```
dataporter.config=INFO
dataporter.statistics=INFO
```

MicrosoftLogin_clientSecret

Client Secret is `Client Secret` provided by Microsoft when you create an Application `Credentials & Secrets` in Microsoft.

NevisAdaptAnalyzerConfig_ipAnalyzer

Used to disable IpAddress analysis. If you wish to disable filtering for private address, the configuration can be found at `nevisAdapt Instance / IP Geolocation` .

If you wish to disable this setting also consider disabling the IP Geolocation settings as well in the `nevisAdapt Instance / IP Geolocation` configuration and the `nevisAdapt Instance / IP Reputation` configuration.

SamlIdp_authenticationType

Select which authentication types are allowed:

- `SP-initiated` : recommended
- `IDP-initiated` : less secure as no `AuthnRequest` is sent.

NevisAdaptLogSettings_serverSyslogFormat

Logback log format for the SERVER SYS logs.

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

CustomAuthLogFile_maxBackupIndex

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

NevisIDMUpdateUserLoginInfo_onSuccess

Configure the step to execute after the user's login info is updated. If no step is configured here the process ends with `AUTH_DONE` .

NevisAuthDeployable_threads

Number of threads to process incoming requests.

CustomAuthLogFile_eventLog

Enable event logging capability of nevisAuth.

JWTToken_header

When this pattern is assigned to an application, the JWT token will be added to all requests which are forwarded to that application.

Here you can define the name of the HTTP header which should contain the token.

CustomProxyLogFile_maxBackupIndex

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

SamlIdpConnector_signerTrust

Assign a pattern to configure the signer certificate of the identity provider.

NevisMetaDatabase_schemaUser

The user which will be used to connect to the database and create the schema (tables).

The database must have been created already (`CREATE DATABASE`) and the user must have `CREATE` privileges for this database.

Example: `umet01`

TANBase_level

Set an authentication level if authentication of this step is successful. The level is relevant only if there are is an Authorization Policy assigned to applications.

NevisAuthDeployable_differingResourceStartover

Define the value of the `AuthEngine` attribute `differingResourceStartover`.

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

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

NevisAdaptAuthenticationConnectorStep_onUntrained

Set the step to continue with in case the user is untrained.

Risk Profile configuration: Setting this step is optional, but the highest available from High and Medium step will replace it.

Risk Event configuration: Setting this step is mandatory.

NevisAdaptAnalyzerConfig_geoIpAnalyzer

Geo/IP Analyzer is a global setting, disabling this means that the device analyzer will not be used to calculate risk scores. This will result in a lower risk score for all users.

If you wish to disable, consider disabling all other submodules as well.

SamlIdpConnector_nextSteps

Assign follow-up steps.

The order of steps is relevant. The first step in this list has index `1`.

You may reference a step in the configuration via the `Custom Transitions`.

NevisIDMPruneHistoryJob_retention

Define how long history data shall be kept in days.

Example: `30d`

The minimum value is `1d`. The maximum value is `1024d`.

ApplicationProtectionDeployableBase_trustStore

Reference a trust store provider pattern or leave empty to manage the trust store with nevisAdmin.

SamlSpConnector_multiValue

This setting defines how multi-value attributes are added to the SAML Assertion.

Example for `enabled`:

```
<saml2:Attribute Name="example">
  <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">value 1</saml2:AttributeVal
```

```
    <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">value 2</saml2:AttributeVal
  </saml2:Attribute>
```

Example for `disabled`:

```
  <saml2:Attribute Name="example">
    <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">value 1,value 2</saml2:Attr
  </saml2:Attribute>
```

GenericDeployment_deleteUnknownFiles

If enabled, all files in the directory (`Path` property) that are not specified under `Files` are deleted.

If you enable this property your files must be in one of the following directories or subdirectories:

- `/var/opt`
- `/tmp`
- `/home`

GenericSMTPChannel_smtpPass

If a password is required at the SMTP server enter it here.

SamlSpConnector_subjectConfirmation

Many SAML service providers require a subject confirmation element to be present in the SAML assertion.

Select `bearer` to add a bearer subject confirmation.

Further options may be provided in future releases.

It may be required to set additional properties. Consult the documentation of the nevisAuth `IdentityProviderState` and apply them via `Custom Properties`.

NevisAdaptAuthenticationConnectorStep_onTimeout

Set the step to continue with in case the authentication attempt runs into a timeout.

Risk Profile configuration: Setting this step is optional, but the highest available from High and Medium step will replace it.

Risk Event configuration: Setting this step is mandatory.

NevisMetaServiceAccessBase_backendKeyStore

Assign a key store if you want to use 2-way TLS for the connection between nevisProxy and nevisMeta.

ServiceAccessBase_path

The (base) path of the application.

Examples:

- `/app/` - defines a base path. Any requests which have a path component starting with `/app/` will be sent to this application.
- `/` - forward all requests to this application. Use this only when there are no other applications or hosted resources.
- `exact:/app.html` - matches requests to `/app.html` only (query parameters are allowed). Use this for single-page applications which don't require any additional resources.

Note that if the frontend path is different from the path used within `Backend Addresses` then URL rewriting will be configured to correctly route requests and responses between clients and backends.

AuthCloudLogin_onUserNotExists

Assign an authentication step to continue with when the user does not exist or has no active authenticator.

If no step is assigned here the authentication flow will fail for such users.

NevisAdaptDeployable_port

Enter the port on which nevisAdapt will listen.

NevisFIDODeployable_authorizationDeregistration

TODO

NevisFIDODatabase_encryption

Enables SSL/TLS in a specific mode. The following values are supported:

- `disabled` : Do not use SSL/TLS (default)
- `trust` : Only use SSL/TLS for encryption. Do not perform certificate or hostname verification. This mode is not safe for production applications but still safer than `disabled` .
- `verify-ca` : Use SSL/TLS for encryption and perform certificates verification, but do not perform hostname verification.
- `verify-full` : Use SSL/TLS for encryption, certificate verification, and hostname verification.