Technical Report

Multifactor Authentication in Element

Configuration Guide

Erik Kemp, NetApp
June 2020 | TR-4840

Abstract

This guide describes how to use multifactor authentication with NetApp® Element® software 12.0 or later.

NetApp®

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

This document describes the multifactor authentication (MFA) configuration with NetApp® Element® software 12.0 or later. MFA increases user security beyond the use of passwords alone.

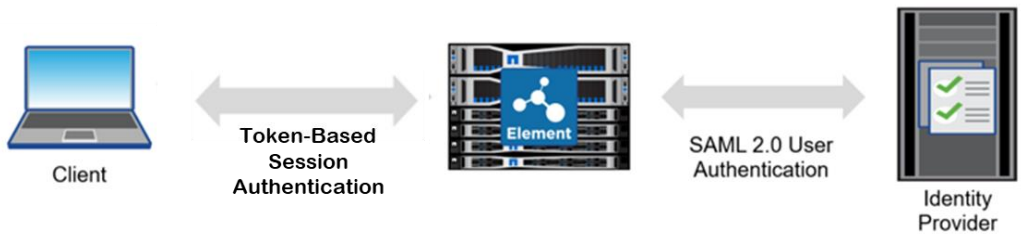
Multifactor authentication features:

- Expands user interface authentication to support MFA on the following user interfaces:
  - Element Cluster UI
  - Per-node UI on Element storage nodes (after cluster creation)
  - NetApp Hybrid Cloud Control manageability suite hosted on an Element management node
- Can integrate user interface login into existing single sign-on (SSO) infrastructure.
- Integrates with industry-standard identity providers (IdPs) via SAML 2.0:
  - Active Directory Federation Services (AD FS)
  - Shibboleth 3.4.4
- Cluster can be converted to use (or not use) MFA without impact to client I/O:
  - Session authentication can be used without MFA

1.1 Overview

According to the National Institute of Standards and Technology, multifactor authentication (or two-factor authentication, 2FA), is a security enhancement that allows you to present two or more pieces of evidence – your credentials – when logging in to an account.

Figure 1) Multifactor authentication overview.



2 Authentication Methods and Admin Types

2.1 Authentication Methods

Table 1 shows user access methods for MFA.

Table 1) Supported authentication methods for each user access method.

Authentication Method	Direct API Calls[1]	User Interfaces[2]
Basic Auth	Yes	No
IdP Auth	No	Yes
Session Auth	Yes[3]	Yes

Authentication Method	Direct API Calls[1]	User Interfaces[2]
[1]Direct API call examples: Using JSON-RPC via a browser, curl, etc. [2]Supported user interfaces: Element UI, Node UI, and NetApp HCI Management UI/Hybrid Cloud Control [3]This functionality is somewhat limited. Script writers must make sure that the script talks with the Element authentication container endpoints to generate and refresh the short-lived tokens.		

You may want to configure your cluster to support MFA for UI access while allowing direct API calls via scripts that use Basic authentication.

Table 2 outlines what is allowed when IdP and LDAP (or local) cluster admin users are configured.

Table 2) Supported admin users and HTTP authentication method with cluster authentication modes.

Admin Type and HTTP Authentication Method	LDAP Disabled IdP Disabled	LDAP Enabled IdP Disabled	LDAP Disabled IdP Enabled	LDAP Enabled IdP Enabled
Cluster Admin Basic Auth API Call	Yes	Yes	Yes	Yes
Cluster Admin Bearer Auth API Call	Yes	Yes	Yes[1]	Yes[1]
IdP Admin Basic Auth API Call (Never Allowed)	No	No	No	No
IdP Admin Bearer Auth API Call	No	No	Yes[2]	Yes[2]
LDAP Admin Basic Auth API Call	No	Yes	Yes	No
LDAP Admin Bearer Auth API Call	No	Yes	Yes[1]	No
[1]These API calls would be made by a script writer using Bearer authentication and not via a supported UI. [2]Authentication for an IdP admin is performed by the identity provider when the user is redirected to the IdP login page by the supported UI.				

3 High-Level Login Flow with MFA

1. User contacts the Cluster UI on MVIP.
2. User is redirected to their IdP for authentication.
3. User is prompted for a second factor, if configured.
4. User is returned to the Cluster UI:
   a. SAML assertion is returned to the cluster.
   b. User authentication is confirmed and authorization is validated.
5. Cluster UI loads and user proceeds normally.

4 Getting started with MFA

4.1 Configuring MFA via API

You can use these basic steps in the Element API to set up the MFA feature. For details of each API method, see the Element API Reference Guide.

1. Connect to IdP and retrieve XML metadata.
2. Escape double quotes to allow metadata to be passed via API:

a. Find and replace “ with \”.

b. JSON formatter tools.

Note: IdP metadata, in plain text format, is retrieved from the third-party IdP. This metadata needs to be "escaped" to be formatted properly in JSON. This can be done with an application like this one: https://freeformatter.com/json-escape.html.

3. Run the `CreateIdpConfiguration` API method, which allows you to create a potential trust relationship for authentication using a third party IdP for the cluster. A SAML Service Provider (SP) certificate is required for IdP communication; the certificate is generated as necessary.
4. Configure IdP with cluster SAML Service Provider metadata:
   a. Retrieve cluster metadata from the URL in step 3.
   b. Configure SAML assertions, on the third-party IdP, to include NameID to uniquely identify a user for audit logging, and for single logout to function properly.
   c. Add a trust relationship: Create IdP cluster admin users by using the `AddIdpClusterAdmin` API method, which adds a cluster administrator user authenticated by a third-party IdP.
   IdP cluster admin accounts are configured based on SAML attribute-value information provided within the IdP’s SAML assertion associated with the user. If a user successfully authenticates with the IdP and has SAML attribute statements within the SAML assertion that match multiple IdP cluster admin accounts, the user will have the combined access level of those matching IdP cluster admin accounts.
5. Create IdP cluster admin users by using the `AddIdpClusterAdmin` API method.
6. Enable MFA on the cluster by using the `EnableIdpAuthentication` API method, which enables support for authentication using a third-party IdP for the cluster.
   Once IdP authentication is enabled, cluster and LDAP admins will no longer be able to access the cluster via supported UIs, and any active authenticated sessions will be invalidated or logged out. Only third-party IdP authenticated users will be able to access the cluster via the supported UIs.
7. Log in to the UI via IdP user on IdP (supply cluster metadata).

5 Workflow for Configuring IdP-Based Authentication with Element

5.1 Set Up NameID Policy for IdP

For auditability of individual user actions on the cluster, NetApp highly recommends a persistent NameID that is unique per user.

- AD FS
- Shibboleth

Note: If you don’t configure and release a persistent NameID, and you don’t release the UID, then logout won’t work correctly from an IdP-authenticated session.

5.2 Create IdP-Based Cluster Admins

1. Create cluster admins based on desired user and/or group mappings that you want to relate to the cluster.

   Note: IdP authenticated users who match multiple Element Cluster IdP administrators will receive the aggregate permissions of those matches.

2. Use the `AddIdpClusterAdmin` API to configure the username for the IdP cluster admin, which should match the SAML Attribute Name/Value mapping for the desired effect. (For security, the SAML Attribute Name/Value pairing is case sensitive.) Examples:
   – email=bob@company.com if the IdP is configured to release email address in the SAML attributes

- group=cluster-administrator if the IdP is configured to release a group property in which all users should have access

5.3 Retrieve the IdP Metadata Contents from the IdP

1. Create an IdP configuration on the Element Cluster with the retrieved metadata: `createidpconfiguration`.

   The metadata content must be passed into the `idpMetadata` parameter and must be URL encoded to comply with the JSON-RPC specification.
2. If the IdP metadata changes (for example, if the certificate is replaced), you will need to retrieve the new IdP metadata, disable IdP authentication on the cluster, upload the new metadata using the `UpdateIdpConfiguration` API call, and finally reenable IdP authentication on the cluster.

5.4 Retrieve the Cluster's Service Provider Metadata

1. Retrieve the `spMetadataUrl` field from the `CreateIdpConfiguration` or `ListIdpConfigurations` API call.
2. Use `spMetadatUrl` to retrieve the SP metadata from the cluster.

5.5 Configure the IdP to Accept the New Service Provider Metadata

- Shibboleth:
  a. Configure Metadata Provider for Cluster SP.
  b. Update attribute filter to release any desired SAML Assertion Name/Value pairs.
  c. Add Relying Party details related to the Cluster SP.
- AD FS:
  a. Configure Relying Party Trust using MS workflow:

     https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/operations/create-a-relying-party-trust

If you later use `UpdateIdpConfiguration` with `generateNewCertificate=true`, you will need to repeat the process of retrieving the SP metadata and configuring the IdP to accept it.

5.6 Enable IdP Authentication

`EnableIdpAuthentication`

Enable support for authentication using a third-party IdP for the cluster.

Once IdP authentication is enabled, cluster and LDAP admins are no longer able to access the cluster via supported UIs, and any active authenticated sessions are invalidated and/or logged out.

Only third-party IdP authenticated users will be able to access the cluster via the supported UIs.

Table 3) Enable IdP authentication.

Parameter	Description	Type	Optional	Visibility
idpConfigurationID	UUID for the third-party IdP configuration. If only one IdP configuration exists, the UI defaults to enabling that configuration.	UUID	True	Public

If something goes wrong, you can disable to revert the UI back to local or LDAP-based cluster admins:

`DisableIdpAuthentication`

1. Disable support for authentication using third-party IdPs for the cluster.
2. Once disabled, users authenticated by third-party IdPs are longer be able to access the cluster, and any active authenticated sessions are invalidated and/or logged out.
3. LDAP and cluster admins are able to access the cluster via supported UIs.

6 Active Directory Federation Services (AD FS)

6.1 To Configure AD FS

1. Download AD FS metadata from the server (https://<server FQDN>/FederationMetadata/2007-06/FederationMetadata.xml).
2. Format XML in cleaned-up markup in a text editor or by using an XML formatting tool like Tidy.
3. JSON escape the XML (https://www.freeformatter.com/json-escape.html).
4. Run `CreateIdpConfiguration`.
5. Retrieve the SP/cluster metadata (https://<MVIP>/auth/ui/saml2).
6. Configure Relying Party Trust (https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/operations/create-a-relying-party-trust):
   a. Follow the default options.
   b. Import metadata from a file.
   c. Do not configure MFA
7. Create Claim Rules:
   a. Custom Rule →
   b. `c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer == "AD AUTHORITY"] => issue(store = "Active Directory", types = ("mail", "memberOf", "sAMAccountName", "userPrincipalName"), query = ";mail,memberOf,sAMAccountName,userPrincipalName;{0}", param = c.Value);`
   c. LDAP (Active Directory as source): User-Principal-Name to UPN.
   d. Transform: UPN to Email Address (Pass through all).
   e. Transform: Email to NameID (Format Email) (Pass through all).
8. Create `IdpClusterAdmin` users:
   a. `userPrincipalName=exampleusername@exampledomain.com`
   b. `memberOf=CN=USStorageAdmins,OU=Groups,OU=US,DC=prodtest,DC=solidfire,DC=net`
   c. `sAMAccountName=exampleadminou00`

6.2 To Use Nested Groups

1. Add these two custom rules:

```
c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer
== "AD AUTHORITY"] => add(store = "Active Directory", types = ("UserDN"), query =
";distinguishedName;{0}", param = c.Value);
c1:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer
== "AD AUTHORITY"] &&
```

```
c2:[Type == "UserDN"] => issue(store = "Active Directory", types = ("MemberOfDN"), query =
"(member:1.2.840.113556.1.4.1941:={1});distinguishedName;{0}", param = c1.Value, param =
c2.Value);
```

2. Add an `IdpClusterAdmin`:

```
MemberOfDN=CN=GlobalStorageAdmins,OU=GlobalGroups,DC=prodtest,DC=solidfire,DC=net
```

Note: This type of `IdpClusterAdmin` can be any of the groups (or nested parent groups) that the user is a member of. This is like the functionality that exists with LDAP users.

7 Notes on Enabling MFA

- When MFA is enabled (`EnableIdpAuthentication` API) all other authentication methods are blocked for the UI (local and LDAP cluster admins).
- When enabling or disabling IdP authentication, all existing sessions are invalidated.
- Local and LDAP cluster admins are still allowed to use the API via Basic authentication.
- If a customer wants to enforce MFA on UI and use scripting, they need to create both IdP cluster admin users and local or LDAP cluster admins (see Table 2).
- If the IdP configuration is broken, disable MFA (`DisableIdpAuthentication`) via the API, fix it, and reenable.

8 Notes on Configuring MFA

- `NameID` must be configured on the IdP; this is how user actions are logged in the event log and syslog. Customers can choose which field is mapped to `NameID`; email address is recommended.
- Be sure to escape the XML from your IdP, or the `CreateIdpConfiguration` API will fail.

9 Notes on Creating IdpClusterAdmin Users

The Username field in the `AddIdpClusterAdmin` API method must match one of the key/value pairs returned by the IdP.

For example:

- name=bob@netapp.com
- uid=user4
- memberOf=ou=storageadmins,dc=domain,dc=com

These attributes are configurable on the IdP. Some are standard, some are not.

10 MFA API Commands

This section lists all of the APIs available for managing and configuring MFA and session management.

10.1 IDP Management

```
CreateIdpConfiguration
UpdateIdpConfiguration
DeleteIdpConfiguration
ListIdpConfigurations
```

10.2 IDP Authentication

```
EnableIdpAuthentication
```

```
DisableIdpAuthentication
GetIdpAuthenticationState
```

10.3 IDP User/Group Management

```
AddIdpClusterAdmin
ListClusterAdmins
ModifyClusterAdmin
RemoveClusterAdmin
```

10.4 Session Monitoring

```
ListActiveAuthSessions
ListAuthSessionsByClusterAdmin
ListAuthSessionsByUsername
```

10.5 Session Management

```
DeleteAuthSession
DeleteAuthSessionsByClusterAdmin
DeleteAuthSessionsByUsername
```

11 Element API Automation with Bearer Tokens

11.1 Overview

Starting with version 11.3, Element software supports using bearer tokens to authenticate API calls in addition to Basic authentication. The addition of bearer token support is accomplished by adding endpoints that support the OAuth 2.0 and OpenID Connect 1.0 specifications for requesting tokens.

11.2 Obtaining a Token

The first step in using Element automation with bearer tokens is to obtain a token. This is done by making an HTTP POST request to the Element token endpoint on the cluster MVIP address. The format of this request is defined by the OAuth 2.0 specification. Here is a sample curl command that requests a token:

```
curl -k -X POST \
 https://<cluster-mvip-address>/auth/connect/token \
 -F client_id=element-automation \
 -F grant_type=password \
 -F username=<cluster-username> \
 -F password=<cluster-password>
```

This example obtains a token by using the `resource owner (password)` grant type, which allows authentication using the username and password for any local or LDAP cluster admin. The client_id of `element-automation` identifies the request for a token that can be used for automation of the Element API. The username and password fields are self-explanatory.

Assuming that all went well, the token request returns a token response. The response is a JSON-encoded object that contains the access token and some metadata, like this:

```
{
    "access_token": "a-huge-string-of-random-encoded-letters-and-numbers",
    "expires_in": 300,
    "token_type": "Bearer",
    "scope": "element_api",
}
```

This (JSON formatted) response contains the token (`access_token`), when it expires (in seconds), what type of token it is, and any scope the token is intended for.

11.3 Using a Token

Once a token has been obtained, it can be added to Element API calls made against the Element cluster. These tokens are valid for both cluster and node API calls, if the node is a member of the cluster that generated the token.

To use a token to authenticate an Element API call, it must be included as a bearer token in the request's Authorization header. Here is an example request:

```
curl -k -X POST 'https://<cluster-mvip-address>/json-rpc/12.0' \
--header 'Authorization: Bearer <access-token-string>' \
--data-raw '{
"id": 1,
"method": "GetClusterInfo",
"params": {
}
}'
```

Note the use of the Bearer type in the authorization header instead of Basic, as would be seen for HTTP Basic authentication.

The same token can be used for multiple API calls, until its expiration.

11.4 Token Expiration

Each token expires after a number of seconds as returned in the `expires_in` field in the response from the cluster's token endpoint. When that amount of time passes, the token can no longer be used for authorizing API calls. A request with an expired token is returned with a 401 Unauthorized response. Any automation that is expected to potentially run longer than the expiration set on the token should plan to handle expired tokens. Examples of possible ways to deal with expired tokens include using a timer to obtain a new token when the current token is close to expiring, or handling the 401 response by obtaining a new token and retrying the failed API call. A small leeway (default of 30 seconds) is implemented to allow for clock skew among systems using tokens for authorization.

11.5 Third-Party OAuth/OIDC Libraries

Element provides a set of industry-standard OAuth 2.0/OpenID Connect endpoints. In addition to the token endpoint used earlier, there is an OpenID discovery endpoint that provides discovery of the authentication service settings needed by various third-party libraries. Although covering those libraries is out of scope for this document, they generally require the URL of the discovery document. That endpoint is located at `https://<cluster-mvip-address>/auth/.well-known/openid-configuration`.

12 MFA Cluster Faults

Here is a list of possible cluster faults with MFA.

12.1 AuthenticationServiceFault

- **Name:** Element Auth container unhealthy
- **Severity:** Error
- **Cause:** Logged when the authentication container cannot start or is continually restarting
- **Recovery steps:** Contact support to intervene

12.2 IdPCertificateExpiration

- **Name:** IdP Certificate Expiration
- **Severity:** 30 days – Warning; 7 days – Error; 3 days – Critical; Expired - Critical

- **Cause:** Logged when the IdP cluster certificate is nearing expiration
- **Recovery steps:** Before the certificate expires, update the expiring certificate by using the `UpdateIdpConfiguration` API method

12.3 MFA Is Incompatible with Certificates <2048 Bits in Length

By default, a 2048-bit SSL certificate is created on the cluster. Users should avoid calling the `SetSSLCertificate` API call and setting a smaller sized certificate.

13 If an IdP Certificate Is No Longer Valid

To refresh IdP certificates that are no longer valid, users must use a non-IdP admin to refresh it by using the `UpdateIdpConfiguration` API, which updates an existing configuration with a third-party IdP for the cluster.

13.1 UpdateIdpConfiguration

Updates an existing configuration with a third-party IdP for the cluster.

Table 4) UpdateIdpConfiguration.

Parameter	Description	Type	Optional	Visibility
generateNewCertificate	If true, generate a new SAML key/certificate and replace the existing pair. **Note:** Replacing the existing certificate disrupts the established trust between the cluster and the IdP until the cluster's SP metadata is reloaded at the IdP. If not provided or false, the SAML certificate and key remain unchanged.	Boolean	True	Public
idpConfigurationID	UUID for the third-party IdP configuration.	UUID	True	Public
idpMetadata	IdP metadata for configuration and integration details for SAML 2.0 single sign-on.	String	True	Public
idpName	Name for identifying and retrieving IdP for SAML 2.0 single sign-on.	String	True	Public

14 Troubleshooting MFA

- **Problem:** Admin is unable to create or update an IdP configuration because the IdP metadata is invalid.
  - Check that the metadata has been correctly copied and pasted from the IdP.
  - Check that the metadata has been correctly JSON-encoded for the API call.
- **Problem:** User is not redirected to the IdP login page after IdP authentication is enabled.
  - Check that the correct IdP metadata was provided via `ListIdpConfigurations`.
  - If incorrect, `DisableIdpAuthentication`, fix the metadata via `UpdateIdpConfiguration`, and then `EnableIdpAuthentication`.
- **Problem:** User can reach the IdP login page but is not able to be authenticated.
  - Check the IdP configuration and make sure that the IdP user has the required authentication credentials.

- **Problem:** User can log in to the IdP login page, but the UI is not authorized to call one or more Element API calls.
  - Check that the username SAML attributes of one or more IdpAdmins matches the IdP user attempting to log in and that the access control levels of the mapped IdpAdmin are appropriate by using `ListClusterAdmins`.
- **Problem:** User is frequently required to reauthenticate.
  - Check that the IdP session length is configured with a sensible value.
- **Problem:** Logout does not work correctly.
  - Check that the IdP is configured to return at least one of NameId or UID.
  - Check that the IdP metadata includes a Single Log Out (SLO_ endpoint.

Appendix A: Further Information

NetApp Element 12.0 User Guide

NetApp Element 12.0 API Reference Guide

Appendix B: Terminology

Active Directory Federation Service (AD FS). Enables Federated Identity and Access Management by securely sharing digital identity and entitlements rights across security and enterprise boundaries. AD FS extends the ability to use single sign-on functionality that is available in a single security or enterprise boundary to internet-facing applications to give customers, partners, and suppliers a streamlined user experience when accessing the web-based applications of an organization.

Identity provider (IdP). A system that creates, maintains, and manages identity information while providing authentication services to relying party applications (AD FS, Shibboleth, etc.) → (Authenticates User)

JSON Web Token (JWT). An open standard (RFC 7519) that defines a compact and self-contained way of securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed.

Multifactor authentication (MFA). A method of confirming a user's identity in which a user is granted access only after successfully presenting two or more pieces of evidence (or factors) to an authentication mechanism.

OAuth 2.0 authorization framework (OAuth). Enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf. This specification replaces and makes obsolete the OAuth 1.0 protocol.

OpenID Connect 1.0. A simple identity layer on top of the OAuth 2.0 protocol. It allows clients to verify the identity of the end user based on the authentication performed by an authorization server, as well as to obtain basic profile information about the end user in an interoperable and REST-like manner. https://openid.net/connect/

Security Assertion Markup Language (SAML). Developed by the Security Services Technical Committee of OASIS; an XML-based framework for communicating user authentication, entitlement, and attribute information. SAML allows business entities to make assertions regarding the identity, attributes, and entitlements of a subject (an entity that is often a human user) to other entities, such as a partner company or another enterprise application.

Service provider (SP): A system that receives and accepts authentication assertions in conjunction with a single sign-on profile [Element Cluster] → (Authorizes User).

Session-based (or token-based) authentication. Session information determines the length of time that a user is allowed access to a user interface. Tokens are used to store subsets of this access information between the client and server applications to manage the sessions.

Shibboleth. A standards-based, open-source software package for web single sign-on across or within organizational boundaries. It allows sites to make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner. Shibboleth software implements widely used federated identity standards, principally SAML, to provide a federated single sign-on and attribute exchange framework. Users authenticate with their organizational credentials, and the organization (or identity provider) passes the minimal identity information necessary to the service provider to enable an authorization decision. Shibboleth also provides extended privacy functionality, allowing users and their home sites to control the attributes released to each application.

Where to Find Additional Information

To learn more about the information that is described in this document, review the following documents and/or websites:

- NetApp Product Documentation
  https://docs.netapp.com

Version History

Version	Date	Document Version History
Version 1.0	June 2020	Initial release

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

Copyright Information

Trademark Information

TR-4840-0620