



User Management in Private and Public Clouds

Implementing secure authentication and identity management in enStratus and your cloud environments that integrates into your enterprise IdM systems

Overview

enStratus provides customers with a variety of options to enable each organization to support its governance requirements relating to user authentication, identity, access control, and life cycle management. enStratus breaks user management into five functionally distinct areas:

- Access control
- Directory Services
- Console authentication
- API authentication
- VM authentication

This document focuses on all areas except API authentication. The enStratus REST API document covers API authentication.

Access Control

User management with enStratus leverages one of the most powerful aspects of the enStratus platform.

The challenge of user management breaks down roughly along two boundaries. The first of which is enStratus console access and the second is access rights to an individual cloud resource such as a server.

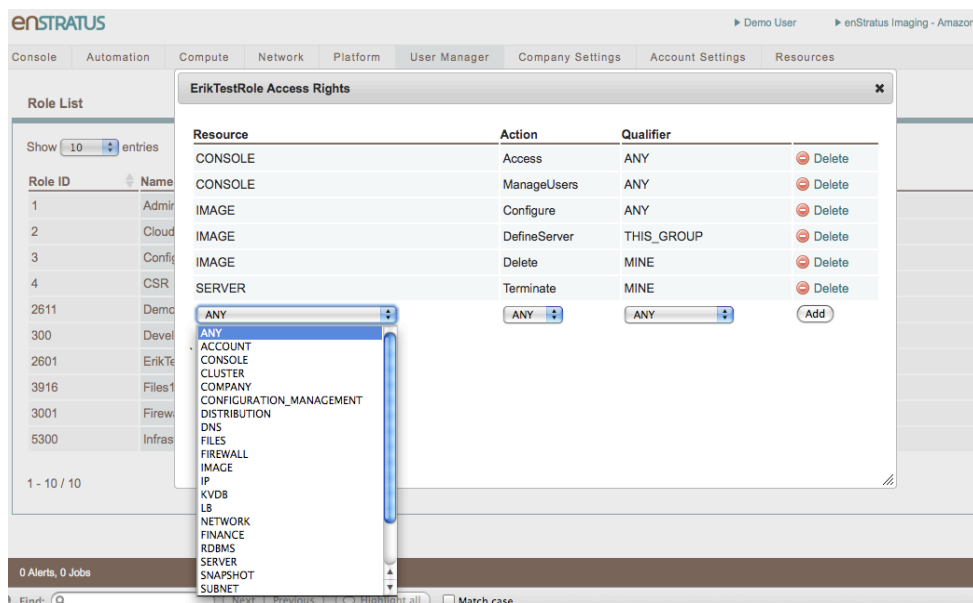
Role-based security allowing users to access or manage resources as required. Users can be alerted to specific actions or issues. Billing codes can be allocated to budget resources.

Roles control user access to the enStratus console. Roles are assigned to Groups in a one-to-one relationship. Each role contains a customizable collection of access rights. Each access right provides access to a certain element of the enStratus console.

RESOURCE: Every page, link and action in the enStratus console is controlled by at least one resource. In most cases, resources correspond to pages. For example, access to the actions available on the Machine Images page is controlled by the IMAGE resource and access to the actions available on the Servers page is controlled by the SERVER resource. Read access and account administration is controlled by the CONSOLE resource.

ACTION: Resources are divided into actions. If you want access to all actions within a resource use the ANY action. If you want the role to have more granular permissions select the specific actions you want users to be able to perform. For example, if you want users with your role to be able to start deployments and services add the CLUSTER-Launch action. Some console actions require multiple resource-action pairs. These are documented in the Important Combinations sections for each resource.

QUALIFIER: There are five different qualifiers: ANY, GROUP, THIS GROUP, BILLING, and MINE. These represent ownership of resources such as servers and machine images. For example, when a user launches a server they can associate it with a group and a billing code. The server that is launched is owned by the group and billing code assigned to it and the user who launched it. With access rights you can limit access to the server to users who belong to the group you chose, the chosen billing code, or the user who launched the server.



Directory Services

enStratus has a logical separation between directory services and authentication to enable greater flexibility for enterprises with existing directory services or identity management systems. In most information security contexts, identity management includes the ability to establish identity. enStratus separates out the storage of identity data from the establishment of identity so you can mix and match what systems represent the authority on who exists (directory services) from what systems assert the identity behind specific operational contexts (authentication).

enStratus currently offers four directory service options:

- Native Directory
- ActiveDirectory
- LDAP
- Auto Provisioning

enStratus minimally requires three pieces of information about an individual to support its operations:

- First name
- Last name
- Email address

enStratus uses email address as an externally identifying field, but retains an internal identifier independent of the email address. Users can also establish other email addresses to use for notifications and such. One primary email address, however, is necessary to enable a user to be authenticated with enStratus regardless of the authentication system in use.

Other values that enStratus leverages that either are not required or are required only in specific contexts include:

- Time zone
- Mobile phone number

enStratus Native Directory

enStratus supports a built-in directory of users and groups that you manage through the enStratus console. A newly created account is initially managed through a native directory. Even if you switch to another option, enStratus under the covers is simply syncing a remote directory service with the native directory.

The enStratus native directory option makes sense for companies that don't have a central directory service or who don't have a complex infrastructure making it difficult for them to add/remove users upon hiring/termination.

LDAP and Active Directory

When you configure enStratus to leverage LDAP or Active Directory (AD) as your authoritative source of user data, you are telling enStratus to synchronize its internal directory with a subset of groups within LDAP or AD and all users in those groups. When you add a user to LDAP, for example, and that user is in one of the synchronized groups, enStratus will discover the user and add that user and all group memberships for that user to enStratus. If that user is terminated, enStratus will deactivate the user in the enStratus directory as well as remove any Windows and Unix accounts for that user on VMs in the cloud.

LDAP/AD integration is an ideal solution with companies with existing LDAP or AD environments and would like to extend their cloud user provisioning and termination policies to the cloud.

There are two mechanisms for setting up LDAP/AD integration depending on your IT policies on exposing this kind of data to third party systems and whether or not you are using the enStratus SaaS or on-premise solutions.

If you are able to provide direct exposure to enStratus from your LDAP/AD server (typical in an on-premise install, atypical for SaaS customers), you configure enStratus to look for a specific set of groups in your directory service. Ideally, you also configure your directory service to limit what enStratus is able to see to just those groups. enStratus will regularly synchronize the users in those groups with the native enStratus directory service. New users will be added to enStratus, terminated users will be deactivated in enStratus, and changes will be pulled from your directory service to enStratus. enStratus does not write to your directory service. enStratus synchronizes only the following user data using LDAP attributes you specify in enStratus:

- First name
- Last name
- Email address
- Mobile phone (optional)
- Time zone (optional)
- Group memberships (only groups in the pre-configured group list)
- It ignores all other information.

For SaaS customers, it's often preferable to setup "staging" LDAP/AD environment in your corporate DMZ. This staging LDAP/AD server will replicate the subset of directory data that should be visible to enStratus and other third parties. enStratus will then synchronize with this staging server instead of direct communication with your LDAP/AD infrastructure.

enStratus talks to ActiveDirectory using the LDAP protocol.

Requirements

- enStratus currently supports only simple authentication with the LDAP/AD server.
- You must know the DN and password for the user under which enStratus will authenticate with the LDAP/AD server.
- You must know the base DN for the location in your directory where users are kept.
- You must know the base DN for the location in your directory where groups are stored.
- You must know what LDAP/AD attributes map first name, last name, email address, mobile phone, and time zone. Mobile phone and time zone are OPTIONAL.

SaaS Configuration

For SaaS customers (on premise customers ignore this), LDAP/AD currently requires a trouble ticket and a services setup fee. Please file a support ticket with support@enstratus.com to initiate this process.

In the future, we will provide a UI for managing this integration.

On-premises Configuration

For on-premises customers, LDAP/AD configuration occurs by installing our directory module in your console network segment. You install this package and then configure the tables `customer_ldap_directory` and `customer_directory_service` in the `enstratus_console` database. You will need the following SQL statements (all of this will soon be configurable via UI):

```
INSERT INTO customer_directory_service ( directory_service_id,
customer_id, active, name, label, description, precedence, type )
VALUES ( 1, CUSTOMER_ID, 'Y', 'My DS Name', NULL, 'My DS Description',
1, 'LDAP' );
```

`CUSTOMER_ID` must match the value of your customer record in your dispatcher database from the `enstratus_customer` table.

```
INSERT INTO customer_ldap_directory ( directory_service_id, customer_
id, customer_admin_group, standard_groups, ldap_access_endpoint, ldap_
access_principal, ldap_access_secret, ldap_access_ssl, ldap_group_base,
ldap_group_description, ldap_group_name, ldap_group_object_class, ldap_
group_usernames, ldap_object_class, ldap_user_base, ldap_user_email,
ldap_user_family_name, ldap_user_given_name, ldap_user_group, ldap_
user_object_class, ldap_user_user_name )
VALUES ( 1, CUSTOMER_ID, ... );
```

The meaning of these values is:

customer_admin_group

The CN of your the group in your directory service (for example, 'Administrators') that will always have administrative access across all accounts in your infrastructure. This value may be null. If null, you must have some non-directory service group as your admin group elsewhere. enStratus will search under `ldap_group_base` for an object with an object class of `ldap_group_object_class` and the CN matching this value. That group will be the admin group.

standard_groups

Similar to `customer_admin_group`, this value is a comma-separated list of group CNs representing the groups to be replicated between enStratus and your directory service. This may be null if you are just replicating the `customer_admin_group`. This list should not, however, include the `customer_admin_group`. Users in these groups will be synchronized into enStratus, but the groups won't have any special access except that defined by roles you assign.

ldap_access_endpoint

For example, `ldap://ad.example.com/`. This is your LDAP server.

ldap_access_principal

The formal DN for the principal being used to authenticate connections with LDAP. For example, 'CN=LDAP User,CN=Users,DC=ad,DC=example,DC=com'.

ldap_access_secret

The encrypted secret password for your directory service access. In order to create this value, you must run the command `java com.enstratus.directory.CustomerLdapDirectory CUSTOMER_ID ENCRYPTION_KEY PASSWORD` and use the encryption key defined in the `classes/cms/networks.cfg` file under the `encryptionKey` value to perform the encryption. You must have the `enstratus-directory-2011.08.1.jar` file in your CLASSPATH. This encrypted value cannot be decrypted with the encryption password alone. The output of the command is what you stick in `ldap_access_secret`.

ldap_access_ssl

A value of 'Y' or 'N' depending on whether SSL should be used for the communications with the LDAP server. 'Y' is recommended.

ldap_group_base

The base DN for the object under which group objects are stored. For example, 'CN=Builtin,DC=ad,DC=example,DC=com'. enStratus will do a deep search under this object for groups.

ldap_group_description

The name of the LDAP attribute containing the value to be used as a group description when the group is represented in enStratus. If you do not have any kind of description attribute, just use the CN value or what is defined in *ldap_group_name*.

ldap_group_name

The LDAP attribute containing the name of the group. This value should be unique across all groups. It is typically 'CN'.

ldap_group_object_class

The object class for groups in your LDAP/AD directory. Typically, this value is 'group'.

ldap_group_usernames

The LDAP attribute that contains the user names of the users that belong to this group.

ldap_object_class

The LDAP attribute that defines an object's object class. For example, 'objectClass'.

ldap_user_base

The base DN under which users are stored. For example, 'CN=Users,DC=ad,DC=example,DC=com'. enStratus will do a deep search under this object for users.

ldap_user_email

The LDAP attribute representing a user email. In some organizations, the email address may be embedded inside a more complex text attribute. You can use the values *ldap_email_regex* and *ldap_email_regex_index* to pull the email value from that text. enStratus requires user emails to be unique and serves as the user's primary mechanism for identifying themselves to enStratus.

ldap_user_family_name

The LDAP attribute that stores a person's last name/family name.

ldap_user_given_name

The LDAP attribute that stores a person's first name/given name.

ldap_user_group

The LDAP attribute that defines the user's group membership.

ldap_user_object_class

The LDAP object class for users. For example, 'user'.

ldap_user_user_name

The LDAP attribute that stores the user name a person uses to login with other systems when a user name (as opposed to an email address) is used to identify them.

enStratus also optionally looks for the following attributes:

`ldap_default_phone_region`

The default region for interpreting the phone number fields. For example, 'US'.

`ldap_email_regex`

A Java regex that tells enStratus how to parse the values in the `ldap_user_email` field.

`ldap_email_regex_index`

The index identifying which part of the regex stores the email address.

`ldap_user_mobile`

The LDAP attribute containing the user's mobile phone number. This value is required when using LDAP with an SMS-based enStratus multi-factor authentication. If the mobile phone number is from a country different than the one defined by `ldap_default_phone_region`, it should start with a country code. Otherwise, enStratus will interpret it as belonging in the default phone region country.

`ldap_user_time_zone`

The LDAP attribute storing the user's preferred time zone.

Auto Provisioning

Auto-provisioning is an option currently available only to enStratus on-premise deployments. In an enStratus on-premise deployment, you can define a site-wide authentication option that points to an enterprise Identity Management (IdM) solution and auto-provisions users into the enStratus directory service based on assertions made by the Identity Management system.

Auto-provisioning only makes sense for customers with an existing IdM solution and an on-premise enStratus deployment. Note that under auto-provisioning, enStratus never knows when a user should cease to have access to VMs in the cloud.

A complete understanding of auto-provisioning requires an understanding of enStratus third-party authentication options. In short, when using a third-party IdM solution, unauthenticated users are re-directed to login window hosted by the IdM solution. The user logs in and then is re-directed back to enStratus. Under auto-provisioning, enStratus will create that user in its native directory service if that users doesn't yet exist.

To set up auto-provisioning, you must first configure enStratus to integrate with your IdM solution (described later in this document).

enStratus has two options related to auto-provisioning that your configure in your `frontend_configuration` table:

- `auto_provision_users`
- `default_group_ids`

To enable auto-provisioning, simply set the `auto_provision_users` value to 'Y'.

Group membership is trickier. enStratus will automatically create groups for any groups provided in assertions from the IdM solution and then place auto-provisioned users in those groups. In most cases, however, the IdM system does not pass group membership information with assertions. A user in no groups essentially has no access to enStratus. You therefore have the option of configuring the `default_group_ids` field in the `frontend_configuration` table as a comma-separated list of enStratus IDs for groups in the enStratus native directory service. All auto-provisioned users are automatically added to these groups.

If you configure default groups, you should remember that anyone who can login to your IdM infrastructure will be able to login to enStratus and be part of those groups. As a result, the default groups should have only the minimal set of permissions necessary to login and do whatever you consider basic work in enStratus.

Authentication

enStratus provides a rich set of authentication options capable of supporting any enterprise authentication requirements, integrated with almost any Identity Management (IdM) system. Specifically, enStratus provides five different authentication options with the ability to add multi-factor (MFA) support onto two of them:

- Native password authentication (managed)
- OpenID authentication with white lists (delegated)
- SAML 2.0 authentication (delegated)
- LDAP authentication (managed)
- enStratus SSO authentication (delegated)

These options fall into two groups: enStratus managed and delegated. A managed option is one in which enStratus provides the login screen and talks to a backend authentication system. A delegated system is one in which enStratus delegates the entire authentication process to an external login page.

Multi-factor may be added on to all enStratus managed options. Whether the delegated options are multi-factor is entirely up to the delegated system. MFA consists of one authentication scheme such as a user name/password pair enhanced with a second factor of authentication such as a one-time password (OTP) sent to the user via SMS or phone call. Because it has a second factor above the the core option, it is inherently more secure than just a single factor.

The natural question here is which option is “the best”. There really is no answer to that question. The best solution depends on whether you are currently using an IdM solution, whether SSO is important to you, and what level of authentication your IT governance policies require. The only general rule is that an MFA is option is (pretty much) always the more secure option than non-multi-factor.

The ability to make definitive statements ends there. Because the delegated options are external, individual implementations may be more secure or less secure than the two managed options. At most, you can comment generally on the assertion mechanisms behind the external protocols: SAML is the most secure, followed by the enStratus native and LDAP, followed by OpenID. The real security value, however, lies in the IdM systems supporting these protocols.

Without going into too much detail at this point, the general recommendation is to use SAML 2.0 with a SAML 2.0-capable IdM solution if SSO matters to you, otherwise use native or LDAP with MFA enabled. The devil, however, lies in the details.

Native Password Authentication

Native password authentication is the basic authentication system we’ve come to expect from Internet-based systems. enStratus requires an email address as the login ID with a strong password to verify the user is who they claim to be. enStratus enables you to define your organization’s policies around password management, including:

- Strength (default: 8 characters with 1 lower, 1 upper, 1 non-alpha, and 1 non-alphanumeric)
- Password rotation (default: no rotation requirements)
- Password history (default: no history is kept, no history is matched during changes)
- Strikes (default: 3)
- Lockout period (default: 30 minutes)

Non-MFA native password is the most reliable authentication mechanism available. All other authentication systems (including MFA) require the involvement of a third-party system in the authentication process. As a result, you can become locked out of your enStratus account if that third-party becomes unavailable. On the other hand, native password authentication lacks the security controls potentially available from your IdM provider. Native password authentication works best for those organizations who value simplicity or availability over the enhanced security of MFA and lack SSO requirements.

enStratus stores all passwords (including password history) as a one-way hash based on user-entered information. It is therefore impossible for enStratus to recover lost passwords. We are working on a simpler mechanism for resetting user passwords. Today, however, a user must contact their enStratus admin to reset a password.

LDAP Authentication

LDAP authentication works exactly like native password authentication, except enStratus calls an LDAP authenticate using the given email address and password instead of checking with a local enStratus database. LDAP works only when you are using an LDAP directory as your directory service.

LDAP authentication is useful when you want to use the same passwords to login to enStratus as with other LDAP-backed systems. LDAP authentication is NOT SSO. Signing into enStratus requires the same password, but signing into enStratus does not sign you in for the other systems you use. To use LDAP authentication with the standard SMS-based MFA, you must have your mobile numbers stored as an attribute of your user objects in your LDAP directory.

OpenID Authentication

OpenID is a federated authentication standard that enables your users to have a single ID managed with an OpenID provider that provides applications like enStratus with assertions related to an individual requesting access to enStratus. For example, user A might be stored with Verisign and User B with Google. Verisign will log user A in and tell enStratus that it vouches for the identity of user A. The same for Google and user B. But Verisign knows nothing about user B and Google nothing about user A.

OpenID is a very low-cost mechanism for setting up a federated identity system. This can be especially good for small/medium sized businesses that need to support many users outside their own organization in their web applications like enStratus. The challenge with OpenID is that you have no control over the password management and authentication policies employed by the OpenID identity provider. enStratus supports whitelisting which can mitigate some of this lack of control.

enStratus supports OpenID with whitelisting. In other words, you can have a traditional OpenID implementation that will talk to any OpenID identity provider or you can define a whitelist requiring all of your users to be managed in one of the whitelisted identity providers.

Configuring OpenID is simply a matter of selecting OpenID as your authentication method and specifying a whitelist, if any. When you set up new users in enStratus, you enter their OpenID ID in addition to other user data. enStratus will then direct them to their OpenID provider when they try to login.

SAML 2.0

SAML 2.0 is really nothing more than a markup language for making assertions. In the context of authentication, a SAML 2.0 identity provider will authenticate users for applications like enStratus. An unauthenticated user requests a resource from enStratus, enStratus formats a request, and the SAML identity provider responds to enStratus with a SAML assertion.

SAML 2.0 is the primary standard for single sign-on support among multiple enterprise-class applications. Most identity management software providers support SAML 2.0, and integrating it with enStratus is fairly simple. If you don't have an existing solution, however, implementing SAML 2.0 can be quite complex.

In addition to using a SAML 2.0 identity provider for authentication, on-premise enStratus customers can configure their enStratus environment for auto-provisioning based on SAML assertions. See the auto-provisioning section of the Directory Services chapter for more information on SAML-based auto-provisioning.

To configure SAML in enStratus, you must provide enStratus with two pieces of information:

- Your SAML endpoint
- Your X509 certificate

enStratus uses the HTTP-Redirect binding for SAML 2.0. Your SAML endpoint must be the URL to which enStratus makes assertion requests based on the HTTP-Redirect binding (this also works with HTTP-POST; enStratus does not support any other bindings).

The X509 certificate is the public certificate used in signing SAML assertions. enStratus will validate any assertions against this certificate.

If you wish to restrict requests against your SAML identity provider, you will need to the SAML ACS URL for enStratus. For the SaaS version, this URL is <https://cloud.enstratus.com/servlet/verifySaml.do>. For an on-premise version, it is https://YOUR_ENSTRATUS_HOST/servlet/verifySaml.do.

enStratus minimally requires an email address in your SAML assertion. The email address must either be part of one the NameID fields in the assertion or an attribute value for an attribute named EMailAddress. This field is the only way enStratus knows for whom the assertion is being made.

enStratus does not sign requests.

enStratus SSO Authentication

enStratus SSO authentication is a proprietary protocol you can implement in a custom application you are looking to integrate with enStratus. When a user requests a page from enStratus and enStratus does not know about that user, enStratus will redirect to your application which will then handle authentication and ultimately redirect back to enStratus with an identity assertion.

enStratus SSO authentication exists for quick integration with custom internal applications. It is simple to implement, but it is proprietary and won't extend to third-party applications. A typical use case is where you have an internally-built system that should work as an extension of enStratus for a similar user base as your enStratus users. If you already have SAML or OpenID in your organization, you shouldn't be using the enStratus SSO authentication.

To support enStratus SSO, you must build an action that handles an enStratus authentication request. The request comes through a query to the following URL:

```
http://YOUR_HOST/YOUR_ACTION?id=AUTH_ID&subject=BELIEVED_EMAIL_
ADDRESS&otp=TOKEN&acs=ENSTRATUS_URL
```

The parameters enStratus are:

id

A unique authentication ID that enStratus generates. You will pass it back to enStratus as part of your verification call so enStratus can tie it to the original query.

subject

This value is the email address that enStratus thinks it is validating. It's actually not important that the email address matches the person who authenticates. It's simply a tool you can use to pre-populate a login form if you like. You can ignore it if you like. This value may be empty if enStratus has no idea who the user might be.

otp

A one-time password that enStratus associates with this request. enStratus generates this value.

acs

The URL to call back to enStratus with the response to the authentication request. When your endpoint receives this request, your application will present the user with a login screen if not already logged in. Upon successful authentication, you will redirect the user back to enStratus at the URL specified in the acs parameter with the following parameters:

id

The same ID enStratus sent your application in the authentication request.

subject

The actual email address of the authenticated user. This is not necessarily the same thing as the subject enStratus passed to your application.

signature

A signature validating that the call is a valid response to the original request. The format of the signature is a hash based on a shared key between enStratus and your application using the following algorithm:

```
BASE64(SHA256(ID:OTP:LOWER_CASE(BELEIVED_EMAIL)))
```

For example, if enStratus makes the following request to your application:

```
http://example.com/sso?id=1234&otp=AbcDef&subject=user@example.
com&acs=https%3A%2F%2Fcloud.enstratus.com%2Fservlet%2FverifySso.do
```

You should authenticate whoever the current user is (even if not user@example.com), and then call enStratus at:

```
https://cloud.enstratus.com/servlet/verifySso.do?id=1234&subject=realuser@example.com&
signature=j4bSSRRQGtSnEUm8VlvXaCPYrSQ3ZkHG3hXrsx/tsrs=
```

Where the signature was calculated as (note it uses the user enStratus thought it was validating, not the user that your application actually authenticated):

```
BASE64(SHA256(1234:AbcDef:user@example.com))
```

The example signature assumes the shared key between enStratus and your application is 1234567890ABCDEF.

Upon validating that signature, enStratus will allow the user access as realuser@example.com.

Multi-factor Authentication

The native username/password and LDAP authentication options are considered single-factor authentication. Specifically, you enter one factor of authentication (a password) to prove you are who you claim to be. Multi-factor authentication combines either of the managed authentication option with one (or more) other “factor” of authentication in order for you to prove your identity. A true two-factor system consists of something you know (like a password) combined with something you have. To gain unauthorized access to an MFA system, an attacker needs to steal both knowledge (your password) and something physical from you.

MFA is recommended for most enStratus customers who do not care about single sign-on with other systems. It combines high security with low complexity. If you leverage the default SMS-based MFA, you should remember, however, that enStratus will be unavailable to you if you are in a place where you cannot receive SMS messages or if the enStratus telephony vendor goes down. In addition, SMS-based MFA is not the strongest form of MFA available as a stolen mobile device for a careless user might have the password stored on it. MFA also requires your own SMS gateway account with a provider like Twilio and incurs extra charges from that vendor. SMS-MFA also does not work on board a plane unless your SMS messages travel over the Internet.

enStratus has a pluggable MFA architecture that can support multiple MFA mechanisms. Currently, enStratus ships with support for SMS-based MFA. If you are interested in other secondary factors, please contact enStratus support.

With SMS-MFA, when a user attempts to login to enStratus, they are presented with a field for their password and a second field for an MFA token. enStratus sends a one-time password (OTP) in the form of a 6-digit code to the user’s SMS device (either via text message or as a voice call). If the user fails to properly enter both items, enStratus denies that user access. After 5 minutes or a successful login, the OTP becomes invalid.

To configure MFA, you simply select MFA as your authentication option in enStratus. It’s really important, however, that all users have configured their SMS number in enStratus. Any user without an SMS number configured will be locked out until their SMS number is set.

VM Authentication

The elasticity inherent in cloud computing creates a unique challenge for enterprise shell and remote desktop access. enStratus operates as an arbiter between your authoritative user database and user access to cloud virtual machines. Because of this arbitration, your cloud virtual machines do not need to participate in any kind of trust relationship with a corporate directory service.

enStratus acts as this arbiter by adding individual user accounts to virtual machines based on the user's enStratus access rights to that VM. enStratus maintains a separate set of login credentials for VM access so that corporate passwords are never placed in a public cloud.

Any enStratus user may be granted shell/remote desktop access to a cloud virtual machine if that user has shell access rights to that virtual machine. If you want to grant a user access to a virtual machine, they must have first created Linux (SSH) or Windows (password) credentials in enStratus. You can then select the virtual machine to which they should be granted access and then grant them access to the virtual machine. When you grant that access, a new user account is created on the virtual machine with that user's authentication credentials.

If you remove the user from enStratus, enStratus also removes all virtual machine access as well.

From an automation perspective, you can also define access rights for deployments and tiers so that users are automatically added to certain VMs (like DBAs to all database servers).

enStratus™ is a cloud infrastructure management solution for deploying and managing enterprise-class applications in public, private and hybrid clouds. enStratus has a multi-cloud architecture that provides governance, automation and cloud independence.

We deliver governance for the cloud through a patent-pending security architecture and a powerful management console across all leading public and private clouds including Amazon Web Services, AT&T Synaptic Storage, Bluelock, Cloud Central, Citrix CloudStack, CloudSigma, EMC Atmos, Eucalyptus, GoGrid, Google Storage, Nimbula, OpenStack, Rackspace, ServerExpress, Terremark, VMware and Windows Azure.

Based in Minneapolis, Minnesota, enStratus serves hundreds of organizations worldwide including Korea's largest telecom provider KT, Silanis, SAIC, and The Cloud Security Alliance.

<http://www.enstratus.com> | 612.746.3091 | contact@enstratus.com