Natural

Natural Availability Server

Version 9.3.2

May 2025

ADABAS & NATURAL

Document ID: ATUX-NHA-932-20250505

Table of Contents

Natural Availability Server

Introduction	What is the Natural Availability Server? The concept, the architecture, and the components.
Installing Natural Availability Server	How to install the Natural Availability Server.
Quick start	Minimal settings to run the Natural Availability Server – not for production.
Configuring the Natural Availability Server	A walkthrough of the configuration parameters for Natural Availability Server.
Advanced Configuration	Insight into advanced application configuration.
Useable Properties	A list of useable properties.
Configuring the User Interface	Application configuration properties that are related to the user interface.
Running the Natural Availability Server on Application Server	How to deploy Natural Availability Server on Tomcat® and Wildfly® application servers.
Building a Docker Image	How to create a Docker image for Natural Availability Server.
User Interface Manual	All about the Natural Availability Server user interface for the end user.
File Transfer	File download, upload, and print, using Natural statements.
Troubleshooting	In case you get stuck.

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at **https://knowledge.softwareag.com**.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Introduction

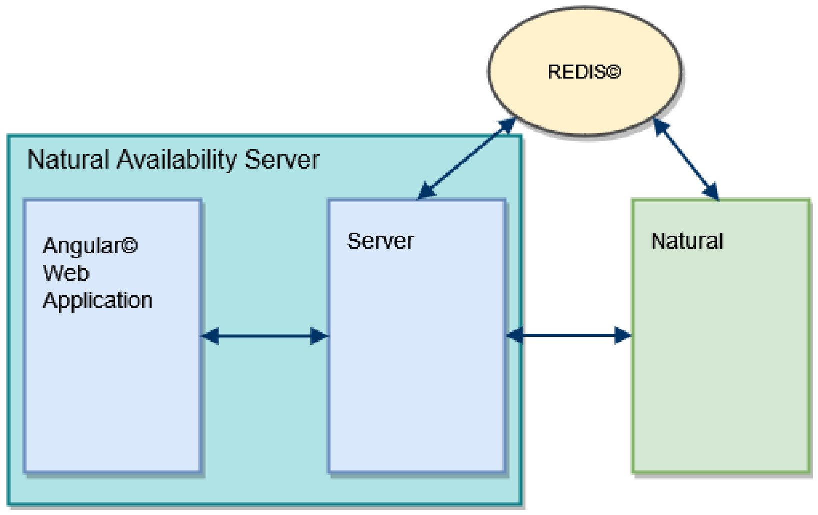
Natural Availability Server is a front-end application that provides a Natural emulator in a modernized Angular© and REST-based web application.

The Availability Server provides capabilities that enable Natural online applications to be operated in a scalable and highly available environment. Based on the availability and SLAs of the underlying infrastructure and the selected architecture, Natural online applications can be configured to achieve system availability of more than 99.99%.

The session context of Natural sessions can be externalized and stored inside a session store, e.g. a REDIS© State Server. This enables the usage of multiple Natural server instances or Linux containers and to place them behind load-balancers. Subsequent requests can be handled by any instance of the Natural Availability Server and any started Natural processes, as session context can be stored and loaded for each request from a distributed clustered in-memory State Server, e.g. REDIS©. For more information on High Availability for Natural, see *Natural/HA (High Availability)*.

Components

The following high-level diagram illustrates the main components of the Natural Availability Server:



The Natural Availability Server has a client-server architecture.

The application is a Web frontend for Natural sessions, that provides High Availability capabilities.

The Server

The server covers following tasks:

- Maintaining a session between the web application and the Natural host
- Synchronizing the data between the web application and the Natural host
- Serving the web user requests and to update the Natural host accordingly

There are two modes to run the server:

- Non-HA Mode
- HA Mode

Non-HA Mode

In this mode the server keeps a session open between the web application and the natural host.

The session is created when the user logs in to the web application and it closes when the user disconnects, or when the program ends.

HA Mode

When HA mode is enabled, each request can be served by any of the servers in the cluster.

All session data is read from the shared session store and stored back for each request.

The Natural session could possibly be closed after each request and a new Natural session will be created for the next request.

Currently, the Natural Availability Server is supporting only REDIS© as a shared session store.

Note: In HA mode, there is an option to set a timeout in the Natural host. The host will wait for a new request before closing the session. This option can be used for performance reasons.

The Client Web Application

The client web application is a modern, web-based frontend that emulates Natural sessions.

More information on configuring the web application behavior can be found in the ***User Interface Manual***.

3 Installing Natural Availability Server

Installer, License, Directory

Natural Availability Server is installed using the Software AG installer and is part of the Natural Product suite.

The installer for Natural Availability Server can be found under **Natural Products** in the installation tree.

The product requires a specific license file to be provided as part of the installation process. If you don't have one, please contact the Software AG sales department.

After Natural Availability Server is installed, a directory */NaturalAvailabilityServer* is placed under the */Software AG* installation directory.

Installing the Natural Availability Server as systemd service

To start the Natural Availability Server during the Operating System's startup:

1. Copy the file *sag_nha.service*, located in the directory */NaturalAvailabilityServer/INSTALL*, to the directory */usr/lib/systemd/system/*.

   Use the following command:

   ```
   sudo cp /NaturalAvailabilityServer/INSTALL/sag_nha.service ↩
   /usr/lib/systemd/system/sag_nha.service
   ```

2. Enable the service:

   ```
   sudo  systemctl enable sag_nha
   ```

4 Quick Start

Note: This section is for running the Natural Availability Server on an embedded Tomcat server. It should not be used for production.

After Natural Availability Server is installed, a directory */NaturalAvailabilityServer* is placed under the */Software AG* installation directory.

Follow the steps below to run the Natural Availability Server after installation.

1. Configure the connectivity to Natural (WebIO server) in the *application.properties* file under */NaturalAvailiabilityServer/conf*.

   In case the default connectivity parameters need to be adapted to your environment, refer to the explanation in the section ***Configuring Connectivity to Natural Host*** to learn how to update the *application.properties*.

2. Configure SSL-related parameters in the *application.properties* file:

   - For connection with SSL, set the following properties:

     ```
     com.softwareag.natural.chimera.connection.ssl=true
     com.softwareag.natural.chimera.connection.certificatePath[0]=<cert path>
     ```

     For details, see section ***Configuring Connectivity to Natural Host***.

   - For a connection without SSL, set the following properties:

     ```
     com.softwareag.natural.chimera.connection.ssl=false
     ```

     Caution: For development only. Not recommended for production!

3. Run `startup.sh` from the directory */NaturalAvailabilityServer/bin*.

These steps will run the Natural Availability Server using its embedded Tomcat®.

To connect to the Natural program from the Angular© web application browse to *http://localhost:8080*

5 Configuring the Natural Availability Server

This section covers all you need for a complete configuration of Natural Availability Server.

Basic Configuration

After installation, the *application.properties* file will be located under the following directory:

<installation location>/NaturalAvailabilityServer/conf

This file supplies all the details required for running the Natural Availability Server, including the host's connectivity data, the type of sessions to be created (HA or not), and all other configurations.

- Configuring Connectivity to Natural Host
- Configuring Connectivity to Natural Using SSL
- Setting the Logging Properties
- Setting the License Location
- Setting the Availability Mode
- Setting the REDIS© Session Store Properties

Configuring Connectivity to Natural Host

The Natural Availability Server can connect in a secure way via SSL protocol, or in a less secure way without SSL.

Caution: The non-SSL connection is not recommended for production use and should only be used for development.

To connect to the Natural host, configure the connection parameters:

Property name	Description	Required	Default and Optional Values
`com.softwareag.natural.chimera.connection.hostname`	Hostname/IP address where the Natural instance is running	No	localhost
`com.softwareag.natural.chimera.connection.port`	Port the Natural Web I/O Interface daemon is listening on	No	2900
`com.softwareag.natural.chimera.connection.ssl`	Define if the Natural Web I/O Interface daemon is using an SSL connection	No	False
`com.softwareag.natural.chimera.connection.parameters[0]`	Define the parameters to start the Natural application Could be more than one parameters	No	Empty value

Property name	Description	Required	Default and Optiona Values
	The next parameter should start using ‘`parameters[1]`’ then ‘`parameters[2]`’ and so on		
`com.softwareag.natural.chimera.connection.certificatePath[0]`	Path for the server certificate used by Natural Web I/O Interface daemon Location: *$NWODIR/$NWONODE/server.cert.crt* of the Natural Host machine	Only for SSL connection	-
`com.softwareag.natural.chimera.connection.application`	The Application script to call to start the session.	No	nwo.sl

Configuring Connectivity to Natural Using SSL

For quick starting out-of-the-box, the Natural Availability Server is configured by default to use non-SSL communication to the Natural Web I/O server.

However, if you want to connect using SSL communication you need to explicitly enable SSL in the following property and change the prarameter to `true`.

```
com.softwareag.natural.chimera.connection.ssl=true
```

The public certificate of the Natural Web I/O server needs to be available to enable SSL communication. This value is configured in the following property:

```
com.softwareag.natural.chimera.connection.certificatePath[0]
```

Note that a fully valid certification path is required to create a secure connection to the host. If the certificate used in the Natural Web I/O server has an unknown intermediate certificate, it should also be defined in the certificate path by assigning a different index, as in the following example:

```
com.softwareag.natural.chimera.connection.certificatePath[1]
```

Setting the Logging Properties

These are the properties that are used to define the log level for each section of the application:

Property name	Description	Required	Default and Optional Values
`logging.level.root`	Default log level for the Application		INFO
`logging.level.org.springframework.web`	Log level for Web requests		INFO
`logging.level.org.springframework.security`	Log level for Spring security		DEBUG

In addition to the properties that can be defined in the *application.properties* file, you can customize the logging output by updating the file *logback-spring.xml* located in the directory *<installation dir>/NaturalAvaialabilityServer/conf*. For more information about this configuration please visit ***https://logback.qos.ch/manual/index.html*** site and follow the configuration link.

Setting the License Location

The following property defines the location of the license file:

```
com.softwareag.natural.web.frontend.server.config.licensePath
```

This property expects the fully qualified or relative path to the file.

In case this property is not set, there are optional possibilities to place the product license file:

1. If the `SAG` environment variable is defined, the application will try to find the license file under:

   ```
   $SAG/common/conf
   $SAG/NaturalAvailabilityServer/conf
   ```

2. If these two directories do not contain the license file, the application will attempt to find the license file in the classpath.
3. The server will stop with an error message if the license file is not found.

Commonly, the name of the expected license file is *nha<version number>.xml*.

However, this name is not used if the license property is set in the *application.properties* file.

Setting the Availability Mode

The following property sets the mode in which the Natural Availability Server will run.

`com.softwareag.natural.web.frontend.server.config.sessionStoreType`

There are two modes available:

1. `memory`
   This is *NON-HA* mode.

   `memory` indicates that the session data is saved in the memory and not in a session store.
2. **redis**

   This is *HA* mode.

   `redis` indicates that the session data is saved to the Redis@ session store.

The following section will explain how to define the properties if option REDIS is selected.

Setting the REDIS© Session Store Properties

The following properties allow the configuration of the REDIS© service location, access credentials, and the use of SSL.

Property name	Description	Required	Default and Optional Values
`spring.data.redis.host`	Host name for the REDIS© server	Required if sessionStoreType is defined as REDIS	
`spring.data.redis.port`	Port on which the REDIS© server is listening	Required if sessionStoreType is defined as REDIS	
`spring.data.redis.ssl.enabled`	Set to `true` if the REDIS© instance is using an SSL connection	Required if `sessionStoreType` is defined as REDIS	
`spring.data.redis.username`	The username that should be used to connect to REDIS©	Required if sessionStoreType is defined as REDIS	
`spring.data.redis.password`	Password to connect to REDIS© if defined See also section ***Encrypting Passwords*** below	Required if sessionStoreType is defined as REDIS	

6 Advanced Configuration

Encrypting Passwords

Each password parameter should be encrypted in the application.properties file.

Procedure

Follow these steps to encrypt a password:

1. Decide on which password to encrypt.
2. Run `encryptPassword.sh` under the */bin* directory with the following command:

   ```
   encryptPassword.sh --encryption-password <JASYPT_ENCRYPTOR_PASSWORD> <passphrase>
   ```

   The `JASYPT_ENCRYPTOR_PASSWORD` property is the key to encrypt the passphrase. It can be any string combination.
3. Place the value received from running the shell script as the password value. This value should be wrapped in ‘`ENC()`’ string to mark the fact that this password is encrypted.
4. In a running testing or production environment, the `JASYPT_ENCRYPTOR_PASSWORD` should be set as an environment variable.

Example

In the following example, the required password for the `spring.data.redis.password` property is ‘`pass1`’, and the `JASYPT_ENCRYPTOR_PASSWORD` property was decided to be ‘`pass2`’.

Run the shell script as follows:

```
encryptPassword.sh --encryption-password pass2 pass1
```

The output received from the shell script could be `AAAABBBB44444FFFBB`.

This value should be set as property in the *application.properties* as follows:

```
spring.data.redis.password=ENC(AAAABBBB44444FFFBB)
```

This process must be done for each password used in the *application.properties* file.

The same `JASYPT_ENCRYPTOR_PASSWORD` property must be used for all encrypted properties.

When subsequently accessing the server that runs the application (for production or testing), an environment variable must be set for the key `JASYPT_ENCRYPTOR_PASSWORD` with the value `pass2`.

Configuration File Option: Class Path

An alternative option to supply the configuration files (*application.properties* and *license file*) is by adding a resource directory to the `NaturalAvailiablityServer` class path in the servlet container:

- For WildFly®
- For Tomcat®

For WildFly®

A module needs to be defined in order to add filles to the class path:

1. Create a directory that contains the resource files.
2. Run the following command to add the module. The module name should stay as defined in the command:

```
/opt/jboss/wildfly/bin/jbos-cli.sh -c -command=”module add
-name=natural-availability-server-configuration --resource-delimiter=,
--resources=<path to folder created>/application.properties, <path to folder
created>/nha<version-number>.xml"
```

For Tomcat®

Configure the location of the *application.properties* file:

1. In the *application.properties*, set the `licensePath` property pointing to the location of the license file.
2. Create the following file under Tomcat® installation:

   <tomcat installation>/conf/Catalina/myHost/NaturalAvailabilityServer.xml

   … where `myHost` is the host name.
3. In this file *NaturalAvailabilityServer.xml*, add the following element:

```
<Context> <Environment name=" spring_config_location" value="<location of
application.properties>/application.properties" type="java.lang.String"/>
</Context>
```

Open Id Connect Authentication

Predefined Public Identity Providers

Support of OIDC (OpenID Connect) authentication requires an identity provider to be defined. Likewise, Natural Availability Server needs to be defined as a client of the identity provider.

Natural Availability Server brings along predefined public identity providers (Google©, GitHub©, and Facebook©).

If you are not using one of those public identity providers, you need to configure a custom identity provider.

Basic Configuration

For the aforementioned commonly used identity providers (Google©, GitHub©, and Facebook©), only a minimal set of properties needs to be defined.

Once these properties are set, the Natural Availability Server will automatically detect that the applicable authentication method is Open Id Connect.

The following minimal properties are required:

Property name	Description	Required	Default and Optional Values
`spring.security.oauth2.client.registration.<identity provider>.client-id`	The client ID issued by the OAuth2 provider.	yes	No default value
`spring.security.oauth2.client.registration.<identity provider>.client-secret`	The client secret issued by the OAuth2. This is a	yes	No default value

Property name	Description	Required	Default and Optiona Values
	sensitive value and should be protected.		
com.softwareag.natural.web.frontend.server.config.oauth2.usernameField	The attribute received from the identity provider that is used for the user name.	yes	email

Note: The `<identity provider>` should be filled with the identity provider name for example `google`, `facebook`, etc.

Optional Properties

The following properties are optional and allow for customizing the user name field and the auto-login.

Property name	Description
com.softwareag.natural.client.config.auto-connect	Define, whe the login to t web applica in OIDC authenticatio will be done automaticall (without activating th Login button the login scr or whether i should be do manually.

Property name	Description	Re
`com.softwareag.natural.web.frontend.server.config.oauth2.removePrefixFromUsername`	Enable removal of the string before the prefix delimiter of the `usernameField` property.	no
`com.softwareag.natural.web.frontend.server.config.oauth2.usernamePrefixDelimiter`	A delimiter that defines the prefix to be removed of the `usernameField` property.	no
`com.softwareag.natural.web.frontend.server.config.oauth2.removeSuffixFromUsername`	Enable removal of the string after the suffix delimiter of the `usernameField` property.	no
`com.softwareag.natural.web.frontend.server.config.oauth2.usernameSuffixDelimiter`	A delimiter that defines the suffix to be removed of the `usernameField` property.	no

In case the OIDC provider is not predefined in the Natural Availability Server, you can configure your own OIDC provider in the *application.properties* file.

Properties Needed for Custom OIDC Provider

For the configuration of custom OIDC identity providers, the following properties need to be considered:

Property name	Description	Required	Default and Optional Values
`spring.security.oauth2.client.provider.<identity provider>.authorization-uri`	The URI where the OAuth2 Authorization Request is sent. This is the endpoint that the	yes	

Property name	Description	Required	Default and Optional Values
	user will be redirected to for logging in.		
`spring.security.oauth2.client.provider.<identity provider>.issuer-uri`	The URI that identifies the issuer of the OpenID Connect (OIDC) provider.	yes	
`spring.security.oauth2.client.provider.<identity provider>.jwk-set-uri`	The URI to the provider's JSON Web Key Set (JWKS), which contains the public keys used to verify the signatures of the JWT tokens.	yes	
`spring.security.oauth2.client.provider.<identity provider>.token-uri=`	The URI where the OAuth2 Token Request is sent to obtain the access token.	yes	
`spring.security.oauth2.client.provider.<identity provider>.user-info-authentication-method`	Defines how the user information request is authenticated, typically by including the access token in the request.	yes	Possible values are: header form etc.
`provider.<identity provider>.user-info-uri`	The URI where the User Info Request is sent to obtain user information from the OAuth2 provider.	yes	
`provider.<identity provider>.user-name-attribute`	The attribute name used to extract the user's name from the User Info endpoint response.	yes	

Note: The `<identity provider>` should be filled with the identity provider name for example `google`, `facebook`, etc.

Example Configuration:

```
spring.security.oauth2.client.provider.custom-provider.issuer-uri=https://accounts.example.com
spring.security.oauth2.client.provider.custom-provider.authorization-uri=https://accounts.example.com/oauth2/authorize
spring.security.oauth2.client.provider.custom-provider.token-uri=https://accounts.example.com/oauth2/token
spring.security.oauth2.client.provider.custom-provider.user-info-uri=https://accounts.example.com/userinfo
spring.security.oauth2.client.provider.custom-provider.user-info-authentication-method=header
spring.security.oauth2.client.provider.custom-provider.jwk-set-uri=https://accounts.example.com/oauth2/certs ↩
spring.security.oauth2.client.provider.custom-provider.user-name-attribute=id
          ↩
```

Optionally, for the configuration of Natural Availability Server as a client of the OIDC identity providers, the following properties should be set:

Property name	Description	Required	Default and Optional Values
`spring.security.oauth2.client.registration.<identity provider>.authorization-grant-type`	The type of authorization grant used to obtain the access token.	yes	The value should be set to: authorization_code
`spring.security.oauth2.client.registration.<identity provider>.client-authentication-method`	The method used to authenticate the client with the OAuth2 provider.	yes	Common values are: basic post
`spring.security.oauth2.client.registration. <identity provider>.client-id`	The client ID issued by the OAuth2 provider.	yes	none
`spring.security.oauth2.client.registration. <identity provider>.client-name`	user-friendly name for the client. It is primarily used in UI contexts.	no	
`spring.security.oauth2.client.registration.<identity provider>.client-secret`	The client-secret issued by the OAuth2 provider. It is a sensitive value and should be protected.	yes	

Property name	Description	Required	Default and Optional Values
`spring.security.oauth2.client.registration. <identity provider>.provider`	The ID of the OAuth2 provider configuration that this registration corresponds to.	yes	
`spring.security.oauth2.client.registration. <identity provider>.redirect-uri`	The URI where the OAuth2 provider redirects the user after authorization. This should match the one registered with the provider. When using the load balancer, the redirect-uri should be specified to match the load balancer's URL.	yes	When using the load-bala be: <load-balancer-url>/login provider>
`spring.security.oauth2.client.registration. <identity provider>.scope[]`	The scopes that your application is requesting from the OAuth2 provider. Multiple scopes can be specified as ... scope[0], scope[1] etc.	yes	

Note: The `<identity provider>` should be filled with the identity provider name for example `google`, `facebook`, etc.

Example Configuration for okta©:

```
spring.security.oauth2.client.registration.okta.client-id=your-okta-client-id
spring.security.oauth2.client.registration.okta.client-secret=your-okta-client-secret
spring.security.oauth2.client.registration.okta.client-authentication-method=post
spring.security.oauth2.client.registration.okta.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.okta.redirect-uri={baseUrl}/login/oauth2/code/okta
spring.security.oauth2.client.registration.okta.scope[0]=openid
spring.security.oauth2.client.registration.okta.scope[1]=profile
spring.security.oauth2.client.registration.okta.scope[2]=email
spring.security.oauth2.client.registration.okta.client-name=Okta
spring.security.oauth2.client.registration.okta.provider=okta
spring.security.oauth2.client.provider.okta.issuer-uri=https://your-okta-domain.okta.com/oauth2/default
          ↩
```

Example: Google Identity Provider

To set up a minimal configuration for OpenIdConnect with Google©

1
- Connect to Google console at: *https://console.cloud.google.com/*
- Choose **APIs and Services**
- On the left-hand side, navigate to **credentials**
- Click on the client link in the OAuth 2.0 Client IDs table

  Create an **OAuth 2.0 client ID for web application** should this entry not exist

2 Copy the `client-id` and `client-secret`, and paste them into the *application.properties* file:

```
spring.security.oauth2.client.registration.google.client-id=<your client ↩
id>.apps.googleusercontent.com
spring.security.oauth2.client.registration.google.client-secret=<client secret>
```

Note: When using the load balancer, the `redirect-uri` should be specified to match the load balancer's URL:

```
spring.security.oauth2.client.registration.google.redirect-uri=<load-balancer-url>/login/oauth2/code/google
```

Health check HTTP call

This HTTP call checks the availability of the Natural Availability Server and the Natural Web I/O Interface server.

When calling the service, the Natural Availability Server will access the Natural Web I/O Interface server and verify its availability.

The HTTP call is designed to supply the availability state of the node to the load balancer.

The availability check is done continuously in the background and the last state is reflected in the health check URL.

The background task can be configured using the following properties:

Property name	Description	Required	Default and Optional Values
`com.softwareag.natural.chimera.connection. continuousHealthcheck`	Determines whether this service is available or not.	false	false
`com.softwareag.natural.chimera.connection.healthCheckInterval`	Defines the time between requests sent from Natural Availability Server to the Natural Web I/O Interface daemon.	false	5000 (milliseconds)

- To call this API, a request needs to be sent to the following URL:

  <server>:<HTTP port>/healthcheck

  Example:

  http://localhost:8080/healthcheck

- Option responses from calling this API are:

  Response code : 200

  Alive - Indicates that the server and host are available.

Error 404
Indicates that the health check API is disabled.

Error 503
Indicates that the Natural Web I/O Interface daemon is not available.

REDIS© Cluster

When using a single-node REDIS© server, high availability cannot be guaranteed because the REDIS© server can fail. To overcome this problem, REDIS© can use replications.

For replications, Natural Availability Server supports REDIS© Enterprise. The REDIS© Enterprise can be used on-premise or as SaaS.

Use the following properties for the REDIS© configuration:

Property name	Description	Required	Default and Optional Values
`spring.data.redis.cluster.nodes`	A list of REDIS© servers in the format of : `<machine1 name>:<redis port>, <machine2 name>:<redis port> etc`	For REDIS© cluster	none
`spring.redis.cluster.max-redirects`	Number of allowed cluster redirections.	For REDIS© cluster	none

Note: The property `spring.data.redis.cluster.nodes` should not be used together with the property `spring.data.redis.host`.

7 Useable properties

The following is a list of useable properties.

Note: Useable application properties are not discussed in any other chapter of the Natural Availability Server documentation

Property name	Description	Required	Defau Optio
`com.softwareag.natural.chimera.connection.column`	Define the number of screen columns	no	80
`com.softwareag.natural.chimera.connection.row`	Defines the number of screen rows.	no	24
`com.softwareag.natural.chimera.connection.max-retry-per-request`	Defines the number of retries allowed to update screen.	no	3
`com.softwareag.natural.chimera.connection.wait-for-next-screen-timeout`	The max time to wait for the new screen recieved after updating screen.	no	6000 (mill
`spring.servlet.multipart.max-request-size`	Used in data	no	

Property name	Description	Required	Default and Optional Value
	transfer for large file uploads.		
`spring.servlet.multipart.max-file-size`	Used in data transfer for large file downloads.	no	

8 Configuring the User Interface

In addition to the aforementioned, the following properties are also part of the application.properties file.

However, these properties are related to the user interface and the Angular© web application:

Property name	Description	Required	Default and Optional Values
`com.softwareag.natural.chimera.fields.input.trim`	Defines if and how to trim the field's content when the screen is loaded. The `trimChar` property contains the value to trim.	No	Optional values: none (default) Left Right both
`com.softwareag.natural.chimera.fields.input.trimChar`	The character to remove from the field's content. **Note:** When defining a space character,	Yes, if the trim property is set.	

Property name	Description	Required	Default and Optional Values
	please set the value to "\ " (backslash and space).		
`com.softwareag.natural.client.config.selectOnFocus`	Define if focusing on an input field will select the full content of the field.	No	false
`com.softwareag.natural.client.config.themesList`	Define the list of themes available in the client UI. The themes define the look and feel of the application. There is an option to create a new CSS theme file and add it to the list.	No	This is the default theme list: emulator.css theme_dark. css theme_light.css

The following table of properties is also related to the user interface and the handling of HTTP errors.

You can use the properties to configure the following:

- Define which HTTP errors cause the application to resend a failed request.
- Define how many attempts are made for a resend.
- Manage the amount of time between resends.

Property name	Description
`com.softwareag.natural.client.config.http-errors.recoverable.enabled`	Define if resending requests for recoverable HTTP errors is enabled.
`com.softwareag.natural.client.config.http-errors.recoverable.max-attempts`	Define the number of times that the application resends a failed request.
`com.softwareag.natural.client.config.http-errors.recoverable.codes`	Define a list of HTTP error codes that cause the web application to resend the request.
`com.softwareag.natural.client.config.http-errors.recoverable.first-timeout`	Define the amount of time in milliseconds that the client waits before sending the request again for the first time.
`com.softwareag.natural.client.config.http-errors.recoverable.timeout-compute-mode`	Define how to compute the timeout for subsequent HTTP requests.

Property name	Description	Require
`com.softwareag.natural.client.config.http-errors.recoverable.multiplier`	Define the multiplier for compute mode MULTIPLE.	No
`com.softwareag.natural.client.config.http-errors.recoverable.timeout-array`	Define the timeout array for compute mode ARRAY.	No

The following table of properties is related to intermediate screens.

Intermediate screens show when the application is performing a task, for example processing data. The user does not expect to insert any data during intermediate screens. When the user gets to an intermediate screen, the web application should try to check if the server has a new screen. The following properties define how to configure the behavior of such requests.

Property name	Description	Required	Def Opt
`com.softwareag.natural.client.config.intermediate-screen.enabled`	Define if resending requests for intermediate screens is enabled.	No	tru
`com.softwareag.natural.client.config.intermediate-screen.max-attempts`	Define the number of times that the application resends a request for an	No	2

Property name	Description	Req
	intermediate screen.	
`com.softwareag.natural.client.config.intermediate-screen.first-timeout`	Define the amount of time in milliseconds that the client waits before sending the request again for the first time.	No
`com.softwareag.natural.client.config.intermediate-screen.timeout-compute-mode`	Define how to compute the timeout for subsequent intermediate screen requests.	No
`com.softwareag.natural.client.config.intermediate-screen.multiplier`	Define the multiplier for compute mode MULTIPLE.	No
`com.softwareag.natural.client.config.intermediate-screen.timeout-array`	Define the timeout array for compute	No

Property name	Description	Required	Def Opt
	mode ARRAY.		

9 Deploying Natural Availability Server on Application Server

Natural Availability Server supports the following servlet containers:

- Tomcat®
- WildFly®

The NaturalAvailabilityServer.war can be found in the following directory:

/NaturalAvailabilityServer/lib

Follow these steps to run Natural Availability Server on a web server:

1. Configure the *application.properties* file (under *<installation location>\NaturalAvailabilityServer\conf*) following the instructions in the section ***Configuring the Natural Availability Server***.
2. Encrypt the passwords in the *application.properties* following the instructions in the section ***Encrypting Passwords***.
3. Place the *application.properties* (and, if needed, the license file) in a location visible to the NaturalAvailiablityServer.

   There are two alternatives for this step:

   - Add the file(s) to the *.war* before deploying the *.war*:

     Navigate to the folder of the *NaturalAvailabilityServer.war* file.
     Add the following directory structure *WEB-INF/classes* to your current directory.
     Under *WEB-INF/classes* add the configuration files (*application.properties* and *nha<version>.xml*)
     Run the command: `jar -u -f NaturalAvailabilityServer.war WEB-INF/classes/application.properties WEB-INF/classes/nha93.xml` (assuming 9.3 is the version).

   - Or ...

     Follow the instructions described in the section ***Advanced Configuration***.

4. Copy the *.war* to the deployment folder in the web server:

Web Server	Directory
Tomcat®	*/Webapp*
WildFly®	*/standalone/deployments*

5. Start the web server application.
6. The Angular© web application will be available under the following URL:

   http://<server location>:<default port of web server>

10 Building a Docker Image

The docker image can be configured for the supported web servers.

There are a number of supported Servlet containers.

Wildfly®	A Dockerfile that uses a standard Wildfly 29 image with Java17 is provided.
Tomcat®	A Dockerfile using a Tomcat 10 image with Java17 is provided.
Embedded	A Tomcat 10 embeded in the delivered war base image using Java17.

To Create the Docker Image

1 **Build the Image**

The script *createDockerImage.sh* can be used to package the Natural Availability Server into an image.

Use the following command to create the image:

```
./createDockerImage [options]
```

The following options can be called:

```
  --appserver    Name of the application server to use ↩
("wildfly"|"tomcat"|"embedded“) (default "embedded")
  --help         Help for this script
```

2 **Run the Image**

The script *createDockerImage.sh* automatically creates a suitable docker run command. After generating an image successfully, the docker run command will appear at the end of the build's docker log.

To run the newly created Natural Availability Server as a container, a valid Natural Availability Server license is needed. The generated command will expect the file to be present in the directory */licenses*.

Example for version 9.3.1

```
podman run -dt -p 8080:8080 -v ↩
./licenses/nha93.xml:/opt/softwareag/common/conf/nha93.xml:z \
  -v ↩
./embedded/application.properties:/opt/softwareag/NaturalAvailabilityServer/conf/application.properties ↩
natural_availability_server:9.3.1
```

3 **Run the Web Application**

Open a browser on the local machine and navigate to the URL *http://localhost:8080/*.

4 **SSL**

For use with SSL encryption, the configuration files have to be adjusted according to the manuals at *https://docs.wildfly.org/* or *https://tomcat.apache.org/*.

11 User Interface Manual

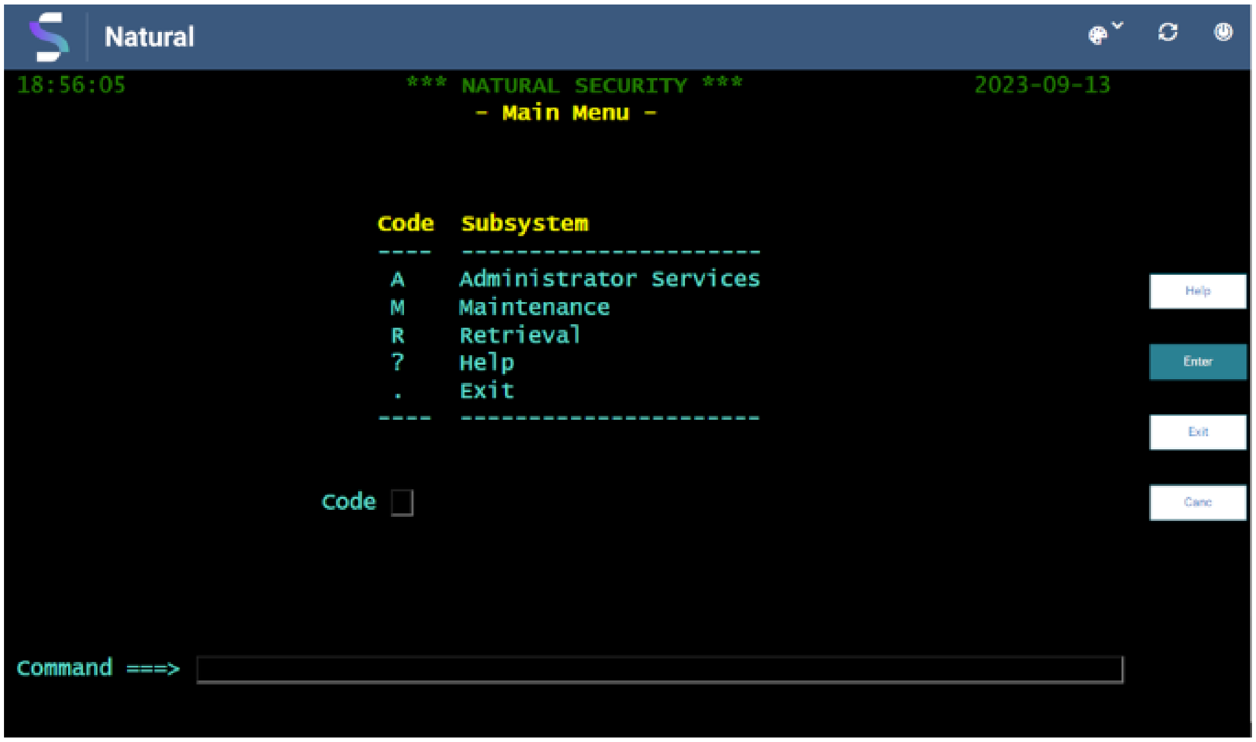
Make sure server is running before accessing the Natural web application. To access the web application, in the default configuration, browse to:

http(s)://<server location>:<default port>

The Natural credentials are required to login to the web application.

The version property in the login screen indicates the current version that you are accessing. If this value is not retrieved, then there is a problem with accessing the Natural Availability Server.

After the credentials are applied, the Natural session starts.



```
18:56:05                     *** NATURAL SECURITY ***                    2023-09-13
                                 - Main Menu -

                         Code  Subsystem
                         ----  ----------------------
                          A    Administrator Services
                          M    Maintenance
                          R    Retrieval
                          ?    Help
                          .    Exit
                         ----  ----------------------

                     Code

Command ===>
```

Example of screen in the Angular web application:

- The main content of the screen is an emulation of the Natural screen.

  It provides input and output fields.
- On the right side of the screen, the current screen's PF keys are displayed as buttons.

  Additionally, each button expands a tooltip that displays the alternative command.
- The top menu shows the following buttons (left to right):
  - **Logoff**

    Closes the current session with Natural
    Redirects the web application to the login screen
  - **Theme**
    Optional look-and-feel flavors to adapt the UI
  - **Refresh**
    Synchronizes the web application content based on the latest state of the Natural session

12 File Transfer

File Download

The Natural Availability Server supports the Natural statement DOWNLOAD PC FILE and transfers the file's content to the browser.

The data file can be downloaded in a format compatible with Entire Connection, or as a CSV file to be opened, for instance, in Microsoft Excel®.

You can configure the default format for the downloaded file via the DOWNLOAD PC FILE statement, using the following properties in the *application.properties* file:

Property name	Description	Required
`com.softwareag.natural.client.config.data-transfer.data-content-file-format`	Defines the format used to save the data file.	no
`com.softwareag.natural.client.config.data-transfer.data-content-expected-behavior`	Defines the handling of the data file.	no

File Upload

The Natural Availability Server supports the Natural statement UPLOAD PC FILE and transfers the file's content to Natural.

The uploaded data file can be in a format compatible with Entire Connection, or as a CSV file that matches the previously downloaded file.

- An upload dialog opens, when the host expects to receive a file.
- Select the file to upload.

Note: The dialog will stay open, if the uploaded file does not match the required format, followed by an error message. You may repeat, and select a suitable file.

Report Printing

The Natural Availability Server supports the statement Natural PRINT and transfers the file's content to the browser to be printed.

The content of the report can be addressed to a printer installed on the client machine or saved to be processed later.

You can configure the default behavior for the PRINT statement using the following properties:

Property name	Descriptior
`com.softwareag.natural.client.config.data-transfer.report-content-expected-behavior`	Defines the handling of the report file
`com.softwareag.natural.client.config.data-transfer.report-content-file-format`	Defines the forma used to save the report file

Automatic Download Property

The browser can handle the download process automatically if a filename is defined in the Natural program using the SET PCFILE command.

To start the download automatically, define the following property:

Property name	Description	Required	Defa and Opti Valu
`com.softwareag.natural.client.config.data-transfer.automatic-download`	Defines whether the download will start automatically, or if a download dialog allows the user to	no	fals (def true

Property name	Description	Required	Default and Optional Values
	configure the behavior for the file.		

Note: This property affects the DOWNLOAD PC and PRINT statements differently.

- DOWNLOAD PC FILE statement

  If the property is set to "true" the download starts automatically.
- PRINT statement

  The content transfers automatically to the client and opens the print dialog.

13 Troubleshooting

Issue:

When connecting to the *URL http://localhost:8080*, I got the message *"This site can't be reached"*.

Solution:

Ensure that ...

… Natural Availability Server is started and listens to the port 8080.
… The firewall is not blocking this port

Issue:

The server did not start and I got the following error in the logs
"redis.clients.jedis.exceptions.JedisConnectionException: Failed to create socket".

Issue:

You have configured the Natural Availability Server to use Redis® as a session store but Redis® cannot be reached.

Check that the following properties are pointing to a working instance of Redis®. You can connect to it from the machine running the Natural Availability Server.

```
# host name for the redis instance
spring.data.redis.host=<your_redis_host_address>
# password to connect to Redis if necessary
#spring.data.redis.password=<xxxxxxx>
# port the Redis server is listening on default is 6379
spring.data.redis.port=6379
# set to true if the redis instance is using SSL connection
spring.data.redis.ssl.enabled=false
# the password that should be used to connect to redis.
#spring.data.redis.password=ENC(encrypted password)
#spring.data.redis.password=clearText
```

```
#the username that should be used to connect to redis.
#spring.data.redis.username=clearText
```

Issue:

When trying to connect to a Natural session, I get the message *"Cannot connect to host"*.

Solution:

Check that the Natural Availability Server can connect to the Natural Web IO (NWO daemon) at the address defined in the following properties :

```
# hostname/IP address where NWO daemon instance is running
com.softwareag.natural.chimera.connection.hostname=localhost
# port the NWO daemon is listening on
com.softwareag.natural.chimera.connection.port=39388
# define if the NWO daemon is using SSL connection
com.softwareag.natural.chimera.connection.ssl=false
# define the parameters to start the Natural application
Com.softwareag.natural.chimera.connection.parameters[0]
```