Administrative access configuration

Administrative user management

This section describes how to set up users and user rights for accessing administrative instruments.

Administrative access to FEJ can be obtained through:

- remote shell
- JMX
- FIXICC app

Configuration of users and their access rights are performed by using Spring Security.

For details on monitoring and management by using a remote shell, refer to FIXEdge Java Administration.

> The changes in the configuration will be applied after FIXEdge Java restart.

Security configuration

FEJ uses Spring Security for authentication purposes. Authentication configuration is located in the **spring/custom-security.xml** file.

File-based authentication

By default, for testing purposes, the FEJ container uses simple in-memory authentication with the `NoOpPasswordEncoder` encoder and plain-text credentials.

For other password encoder options, please check Spring Security 5.0.

If you need a more complex authentication solution, please refer to the Spring Security documentation.

The authentication mechanism is defined in **spring/custom-security.xml**:

```
   <!--Mock Authentication-->
    <sec:authentication-manager id="authenticationManager">
        <sec:authentication-provider>
            <sec:password-encoder ref="passwordEncoder"/>
            <sec:user-service id="userDetailsService" properties="admin-users.properties"/>
        </sec:authentication-provider>
    </sec:authentication-manager>

    <!-- Password encode bean to support plain text passwords in user.properties -->
    <bean id="passwordEncoder"
          class="org.springframework.security.crypto.password.NoOpPasswordEncoder"
          factory-method="getInstance"/>

    <!-- Password encode bean to support passwords encrypted with BCrypt way in admin.properties -->
<!--    <bean id="adminPasswordEncoder"-->
<!--          class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>-->
```

Authorized administrative users are defined in the **admin-users.properties** properties file:

```
# Spring security file format
# password depends on configured spring PasswordEncoder (hash or plain text)
#
# Format: username=password,grantedAuthority[,grantedAuthority][,enabled|disabled]

# password is plain text
admin=admin,JMX_ADMIN,SSH_ADMIN,FIXICC_ADMIN,enabled
guest=guest,FIXICC_GUEST,enabled

# password is hash (bcrypt)
#admin=$2a$10$hCDWIHTwb7zui0dDbG8dXe2r9x3H4JDEynQuoGDn85rk6vOjxGoJC,JMX_ADMIN,SSH_ADMIN,FIXICC_ADMIN,enabled
#guest=$2a$10$pQcJLlpuHRmn5w1MdBx/xudmEBKc0l/ER7TXifc2zntKIrW3lw8S2,FIXICC_GUEST,enabled
```

By default, FEJ provides such user roles with different access level:

- FIXICC_ADMIN - access by FIXICC with ALL permissions (the role is defined in **fixicc_permissions.properties**)
- FIXICC_GUEST - access by FIXICC with ONLY READ permissions (the role is defined in **fixicc_permissions.properties**)
- SSH_ADMIN - access by Remote Shell with ALL permissions (the role is defined in **fixedge.properties**)
- JMX_ADMIN - access by JMX with ALL permissions (the role is defined in **fixedge.properties**)

LDAP authentication

FEJ also supports authentication against an LDAP server.

Before getting deep into LDAP authentication, let's get familiar with some LDAP terms.

Term	Description
Dn	Distinguished name, a unique name which is used to find a user on an LDAP server, for example, in the Microsoft Active Directory.
Ou	Organization unit
Bind	LDAP Bind is an operation in which LDAP clients send bindRequest to an LDAP server including username and password and if the LDAP server is able to find the user and the password is correct, it allows access to the LDAP server.
Search	LDAP search is an operation which is performed to retrieve Dn of a user by using some user credentials.
Root	LDAP directory's top element, like the root of a tree.
BaseDn	Branch in a LDAP tree which can be used as a base for the LDAP search operation.

To activate authentication of administrative users with LDAP, it needs to replace the authentication-manager bean definition in the **spring/custom-security.xml** file:

```
<authentication-manager>
    <ldap-authentication-provider
      ="ou=people"
      user-search-filter="(uid={0})"
      group-search-base="ou=groups"
      group-search-filter="(member={0})">
    </ldap-authentication-provider>
</authentication-manager>

<ldap-server url="ldap://epam.com:389/dc=epam,dc=com"
                    manager-dn="uid=admin,ou=system"
                    manager-password="admin"/>
```

where

Attribute name	Description
ldap-authentication-provider	
user-search-base	Search base for user searches. Defaults to "". Only used with a 'user-search-filter'.

user-search-filter	The LDAP filter used to search for users (optional). For example, "(uid={0})". The substituted parameter is user's login name.
group-search-base	Defines the part of the directory tree under which group searches should be performed. Defaults to "" (searching from the root).
group-search-filter	The filter which is used to search for group membership. Defaults to (uniqueMember={0}). The substituted parameter is the DN of the user.
group-role-attribute	The LDAP attribute name which contains the role name which will be used within Spring Security. Defaults to "cn".
user-dn-pattern	A specific pattern used to build user's DN, for example, "uid={0},ou=people". The key "{0}" must be present and will be substituted with the username.
ldap-server	
url	Specifies the LDAP server URL when not using the embedded LDAP server.
manager-dn	Username (DN) of the "manager" user identity (i.e. "uid=admin,ou=system") which will be used to authenticate to a (non-embedded) LDAP server. If omitted, anonymous access will be used.
manager-password	The password for the manager DN. This is required if the manager-dn is specified.

Check more details about the configuration authentication with an LDAP server on Spring Documentation.

Administrative shell configuration

To configure access to the interactive shell, shell configuration properties are used.

Shell configuration properties are defined by the **shell.properties** file.

Name	Default value	Description
crash.auth	spring	Authentication mechanism
crash.ssh.port	2000	SSH server port to listen
crash.ssh.keypath	/crash /hostkey.pem	The path to the PEM file with an SSH server key. Alternatively, Java Keystore can be used (see the **crash.ssh. keystore.path** option)
crash.ssh.keystore. path		The path to the Keystore file with SSH server keys. Has higher priority than **crash.ssh.keypath**
crash.ssh.keystore. password		The password for the Keystore file
crash.ssh.keystore. provider		Key manager provider. Note that the list of registered providers may be retrieved via the Security.getProviders() method.
crash.ssh.keystore. type	JKS	The type of Keystore. See the Keystore section in the Java Cryptography Architecture Standard Algorithm Name Documentation for information about standard Keystore types. Examples of value: JKS, JCEKS, PKCS12, PKCS11 The default value is JKS (Java Key Store).
crash.ssh.keygen	false	Specify if a key file should be generated during server start. The **crash.ssh.keypath** should be defined.
crash.ssh. auth_timeout	300000	Authentication timeout of the SSH server (in milliseconds)
crash.ssh. idle_timeout	300000	Idle timeout of the SSH server (in milliseconds)
crash.ssh. default_encoding	UTF-8	Character encoding

FEJ uses the Java shell called ‘CRaSH’. For more information about configuration properties, please refer to the CRaSH reference documentation.

Administrative JMX configuration

The **fixedge.properties** file contains settings for defining the JMX port and the URL for accessing the JMX service.

For details, refer to the official Java documentation.

Monitoring and management by using the JMX technology is described in the Management over JMX section.

FIXICC access configuration

For details on the FIXICC access configuration, refer to the FIXICC & FEJ Integration User Guide.

Administrative REST API configuration

FIX Edge Java has REST API functionality that can be used for FIX sessions management.

REST API is configuring in the **fixedge.properties** file.

```
# Enable REST service, if it is false or empty, REST service will not be available
rest.service.enable=true

# Name of REST service, if it is empty or disabled, REST service will not be available and registered in
Service Discovery
# rest.service.name=REST-AdminAPI

# Name of REST port, if it is empty or disabled, REST service will not be available and registered in Service
Discovery
rest.service.port=9010

# Path to SSL certificate, should not be empty to create REST service and register it in Discovery
rest.ssl.cert.path=/ssl/cert.pem

# Path to SSL key, should not be empty to create REST service and register it in Discovery
rest.ssl.key.path=/ssl/key.pem
```