

Configuration of Authentication

- Examples of the Configuration
 - Simple authentication through D2000 users
 - Simple authentication through applicatively defined users
 - Authentication through a D2000 user and a local verification of client certificates
 - Authentication through a D2000 user and a remote verification of client certificates
 - Authentication through applicatively defined users and local verification of client certificates
 - Authentication through applicatively defined users and a remote verification of client certificates
 - Automatic authentication through a predefined D2000 user without a logon dialog box

Configuration of authentication of the SmartWeb application has the following structure in the `smartweb.json` file. the examples of configuration are mentioned below.

smartweb.json

```
{
  /* object with configuration of users' authentication */
  "authentication": {
    "authModes": [
      "AUTH_AUTO_LOGON_IN_SESSION",          // automatic logon to D2000 without display of a
logon dialog box
      "AUTH_CREDENTIALS_IN_SESSION", // name/password authentication to the D2000 user
      "AUTH_CREDENTIALS_IN_RPC",        // name/password authentication through RPC
(applicatively defined users)
      "AUTH_CERTIFICATE_LOCALLY",       // certificate authentication to a local keystore
      "AUTH_CERTIFICATE_REMOTELY",      // certificate authentication through RPC to a
keystore administrated from D2000
    ],

    // D2000 user definition for creating a JAPI session
    // only for the mode AUTH_AUTO_LOGON_IN_SESSION or AUTH_CREDENTIALS_IN_RPC
    "authSessionUsername": "D2000UserName", // user name to D2000
    "authSessionPassword": "D2000UserPassword", // password to D2000

    // path to a keystore and root certificate for validation of clients' certificates
    // only for the mode AUTH_CERTIFICATE_LOCALLY
    "keystorePath": "C:\\path to keystore\\keystore.jks",
    "caCertificateAlias": "SmartWebUsersCert", // alias root certificate to keystore.jks,

    // definition of authentication RPC, only for a turned on mode AUTH_CREDENTIALS_IN_RPC
    "authRpc": {
      "eventName": "E.SMARTWEB_USER",
      "interfaceName": "I.XXX",
      "methodName": "authenticate",
      "useJava": "false"
    },
    "authRpcParams": [ // parameters order of called authentication method, compulsory is only the OUT
parameter _OK (BOOL);
      "USERNAME",
      "PASSWORD",
      "CERTIFICATE",
      "NONE",
      "_OK"
    ],

    // definition of logon RPC method, called automatically after successful authentication, non-
compulsory OUT parameter _OK (BOOL) identifies calling success rate of the logon method;
    "logOnRpc": {
      "eventName": "E.SW_DT_Connect",
      "interfaceName": "I.XXX",
      "methodName": "logOn",
      "useJava": "false"
    },
    "logOnRpcParams": [ // parameters order of the called logon method, compulsory is only the OUT
parameter _OK (BOOL);
      "USERNAME",
      "PASSWORD",
      "CERTIFICATE",
      "NONE",
      "_OK"
    ],

    "logOutRpc": {
      "eventName": "E.SW_DT_Connect",
      "interfaceName": "I.XXX",
      "methodName": "logOff",
      "useJava": "false"
    }
  }
}
```

Examples of the Configuration

Simple authentication through D2000 users

smartweb.json

```
{
  "authentication": {
    "authModes": [
      "AUTH_CREDENTIALS_IN_SESSION"
    ]
  }
}
```

Simple authentication through applicatively defined users

In the following configuration, there is also the 'logOn' method registered due to the identification of [currently logged on user in the called RPC methods](#).

smartweb.json

```
{
  "authentication": {
    "authModes": [
      "AUTH_CREDENTIALS_IN_RPC"
    ],
    // predefined username of D2000 user with whom there will be a session created
    "authSessionUsername": "D2000UserName",
    // predefined password of D2000 user with whom there will be a session created
    "authSessionPassword": "D2000UserPassword",

    // definition of RPC authentication
    "authRpc": {
      "eventName": "E.SW_APPLICATION_AUTH",
      "methodName": "authenticate"
    },
    "authRpcParams": [
      "USERNAME",
      "PASSWORD",
      "_OK"
    ],

    // definition of the logOn RPC method
    "logOnRpc": {
      "eventName": "E.SW_APPLICATION_AUTH",
      "methodName": "logOn"
    },
    "logOnRpcParams": [
      "USERNAME",
      "_OK"
    ]
  }
}
```

Authentication through a D2000 user and a local verification of client certificates

A verification of client certificates takes place locally in the SmartWeb application. All client certificates have to be stored in a keystore by the alias identical with the logon name (beware of the case sensitivity). The certificates have to be signed by the root certificate with a name defined in the `caCertificateAlias` attribute of configuration. For a successful configuration, it is necessary to correctly configure the [element authentication in the standalone.xml file of the Wildfly application server](#). It is required to generate the client certificates according to the steps described in the chapter [Administration of Client Certificates](#).

smartweb.json

```
{
  "authentication": {
    "authModes": [
      "AUTH_CREDENTIALS_IN_SESSION",
      "AUTH_CERTIFICATE_LOCALLY"
    ],
    // path to a keystore with client certificates and a root certificate for validation of client
    certificates: {
      "keystorePath": "C:\\path to keystore\\keystore.jks",
      "caCertificateAlias": "SmartWebUsersCert" // alias of the root certificate in keystore.
    }
  }
}
```

Authentication through a D2000 user and a remote verification of client certificates

A verification of client certificates takes place in D2000 by calling authentication RPC method with a parameter through which there will be the Base64 serialized certificate sent. For a successful configuration, it is necessary to correctly configure the [element authentication in the standalone.xml file of the Wildfly application server](#) where the keystore file has to contain a root certificate with which all client certificates are signed.

smartweb.json

```
{
  /* object with configuration of users' authentication */
  "authentication": {
    "authModes": [
      "AUTH_CREDENTIALS_IN_SESSION",
      "AUTH_CERTIFICATE_REMOTELY"
    ],
    // definition of authentication RPC
    "authRpc": {
      "eventName": "E.SW_APPLICATION_AUTH",
      "methodName": "authenticate"
    },
    "authRpcParams": [
      "USERNAME",
      "CERTIFICATE",
      "_OK"
    ]
  }
}
```

Authentication through applicatively defined users and local verification of client certificates

A verification of client certificates takes place locally in the SmartWeb application. All client certificates have to be stored in a keystore by the alias identical with the logon name (beware of the case sensitivity). The certificates have to be signed by the root certificate with a name defined in the `caCertificateAlias` attribute of configuration. For a successful configuration, it is necessary to correctly configure the [element authentication in the standalone.xml file of the Wildfly application server](#). It is required to generate the client certificates according to the steps described in the chapter [Administration of Client Certificates](#). In the following configuration, there is also the 'logOn' method registered due to the identification of [currently logged on user in the called RPC methods](#).

smartweb.json

```
{
  "authentication": {
    "authModes": [
      "AUTH_CREDENTIALS_IN_RPC",
      "AUTH_CERTIFICATE_LOCALLY"
    ],
    // predefined username of a D2000 user with whom there will be a session created
    "authSessionUsername": "D2000UserName",
    // predefined password of a D2000 user with whom there will be a session created
    "authSessionPassword": "D2000UserPassword",
    // path to a keystore with client certificates and a root certificate for validation of client
    certificates
    "keystorePath": "C:\\path to keystore\\keystore.jks",
    "caCertificateAlias": "SmartWebUsersCert", // alias of the root certificate in keystore.
    jks,

    // definition of authentication RPC
    "authRpc": {
      "eventName": "E.SW_APPLICATION_AUTH",
      "methodName": "authenticate"
    },
    "authRpcParams": [
      "USERNAME",
      "PASSWORD",
      "_OK"
    ],

    // definition of the logOn RPC method
    "logOnRpc": {
      "eventName": "E.SW_APPLICATION_AUTH",
      "methodName": "logOn"
    },
    "logOnRpcParams": [
      "USERNAME",
      "_OK"
    ]
  ]
}
```

Authentication through applicatively defined users and a remote verification of client certificates

A verification of client certificates takes place in D2000 by calling authentication RPC methods with a parameter through which there will be the Base64 serialized certificate sent. For a successful configuration, it is necessary to correctly configure the [element authentication in the standalone.xml file of the Wildfly](#) application server where the keystore file has to contain a root certificate with which all client certificates are signed. In the following configuration, there is also the 'logOn' method registered due to the identification of [currently logged on user in the called RPC methods](#).

smartweb.json

```
{
  /* object with a configuration of users' authentication */
  "authentication": {
    "authModes": [
      "AUTH_CREDENTIALS_IN_RPC",
      "AUTH_CERTIFICATE_REMOTELY"
    ],

    // predefined username of a D2000 user with whom there will be a session created
    "authSessionUsername": "D2000UserName",
    // predefined password of a D2000 user with whom there will be a session created
    "authSessionPassword": "D2000UserPassword",

    // definition of authentication RPC
    "authRpc": {
      "eventName": "E.SW_APPLICATION_AUTH",
      "methodName": "authenticate"
    },
    "authRpcParams": [
      "USERNAME",
      "PASSWORD",
      "CERTIFICATE",
      "_OK"
    ],

    // definition of the logOn RPC method
    "logOnRpc": {
      "eventName": "E.SW_APPLICATION_AUTH",
      "methodName": "logOn"
    },
    "logOnRpcParams": [
      "USERNAME",
      "_OK"
    ]
  }
}
```

Automatic authentication through a predefined D2000 user without a logon dialog box**smartweb.json**

```
{
  "authentication": {
    "authModes": [
      "AUTH_AUTO_LOGON_IN_SESSION"
    ],
    // predefined username of a D2000 user with whom there will be a session created
    "authSessionUsername": "D2000UserName",
    // predefined password of a D2000 user with whom there will be a session created
    "authSessionPassword": "D2000UserPassword"
  }
}
```