



www.rexcontrols.com/rex

OPC UA server for REX

User guide

REX Controls s.r.o.

Version 2.50.5
Plzeň (Pilsen), Czech Republic
2017-09-06

Contents

1	Introduction	3
1.1	OPC UA	3
1.2	OPC UA server for REX	3
1.3	Provided functionality	4
2	Address space	5
2.1	Methods	6
2.2	Blocks	6
2.3	Parameters	6
2.4	Events	7
3	Quick start guide	8
4	Startup and configuration	10
4.1	Startup	10
4.2	System service	10
4.3	Configuration	11
4.4	Configuration sections	12
4.4.1	Target	12
4.4.2	Application	13
4.4.3	Security	14
4.4.4	Auth	15
4.4.5	Endpoint	16
4.4.6	Discovery	17
4.4.7	Options	18
4.5	Configuration templates	21
5	Authentication and authorization	22
5.1	Roles and users	22
5.2	User administration	22
6	Configuration Tool	24
6.1	Certificates	25
6.2	Authentication	27

6.3	Configuration examples	29
7	Connection testing with OPC UA clients	31
7.1	UaExpert	33
7.2	myScada	43
	Bibliography	48

Chapter 1

Introduction

1.1 OPC UA

OPC UA is an open communication protocol for industrial automation. Unlike legacy OPC, OPC UA is a multi-platform protocol, it may work as a web service and it offers many advanced functions like diagnostics, method calls and various levels of security and authentication in addition to standard events and data access. OPC UA is becoming a preferred communication interface of many devices from various companies.

However, OPC UA is not a suitable protocol for hard real-time communication between control devices, but is sufficient for soft real-time applications in many cases. A main utilization areas of OPC UA are human-machine interfaces and interconnection of various devices in a heterogeneous environment.

1.2 OPC UA server for REX

OPC UA server for REX is as standalone application that is connected to a REX runtime utilizing a low-level diagnostic protocol. It is not required to run the OPC UA server on the same station where the REX runtime is running. However, it is advised to run the OPC UA server as close to the REX runtime as possible to minimize latencies. The OPC UA server for REX implements the `opc.tcp` communication protocol, that is the most common protocol for OPC UA servers that acquire data from real-time control devices. The connection of OPC UA with client applications is shown on figure 1.1.

The OPC UA server for REX has to be licensed properly when it operates in an industrial environment. It is also possible to run the server in a so called DEMO mode, in which the server is fully functional but terminates after one hour without any notice. Licensing of the OPC UA server is performed by obtaining a valid license key and registering it either by using `RexDraw` or by setting the `REX_LICENCE_KEY` configuration parameter in the configuration file (see chapter 4.4.2).

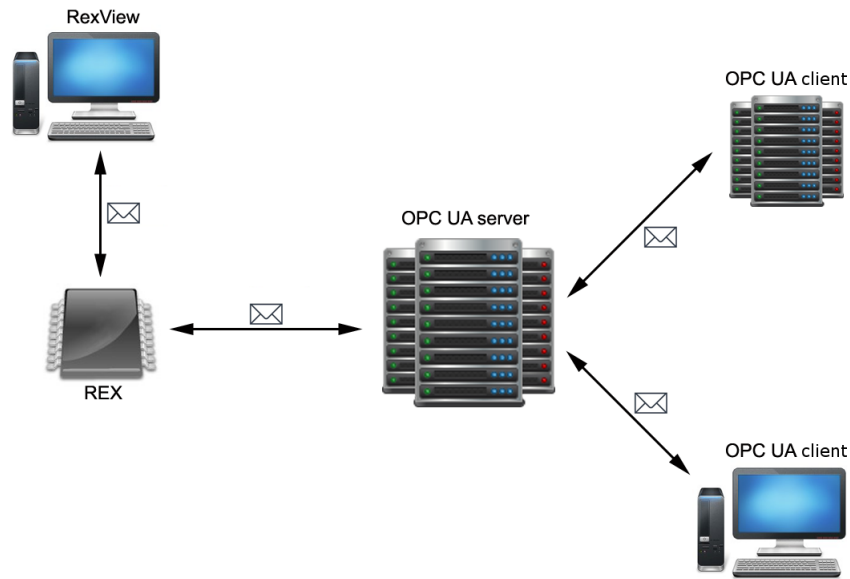


Figure 1.1: Connection between OPC UA server, OPC UA clients and REX runtime

1.3 Provided functionality

The OPC UA server is connected directly to a REX runtime core. It shows complete structure of a target algorithm (ie. blocks, parameters..) in the address space. The server acquires complete structure of the algorithm during startup and make all runtime variables accessible to clients upon request.

A connection to the target device is maintained and checked periodically. The server tries to reinitialize the connection or reconnect to the target device when the connection is lost or an error occurs. Last acquired values are held and available to clients. Value quality is set appropriately. If the target algorithm is changed, the address space is rebuild appropriately and connected clients are notified.

Chapter 2

Address space

An address space contains all the data that is available to clients. Address space is comprised of nodes. Some nodes are common to all OPC UA servers, other nodes are application-specific. There are also nodes that control the server itself. The "Exec" node is a root node to all runtime-specific variables and the whole structure of a target algorithm (ie. tasks, subsystems, blocks and parameters) is available underneath this node. A content of the "Exec" node is rebuilt when a connection with a target is established or a control algorithm is changed on the target. A sample address space is shown on figure 2.1. The picture has been taken from UaExpert OPC UA client (see chapter 7.1).

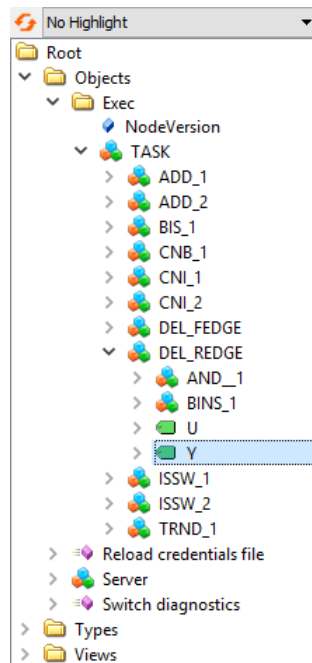


Figure 2.1: Address space in UaExpert

The server utilizes several name spaces. The first name space corresponds to the application URI and is dedicated for a server diagnostics.

The name space *urn:Rex:TypeDeclaration* is dedicated for definitions of types that are used among the address space.

The name space *urn:Rex:Server* is dedicated for commanding the server itself.

A name space that corresponds to the target algorithm is always target-specific and is described in chapter 4.4.2. This name space contains all nodes that corresponds to the target algorithm.

2.1 Methods

The server provides a so-called methods, that may be used to control server's behavior. Administration privileges are required if these methods are invoked by a client.

The method **Reload credentials file** orders the server to reload its configuration file and file with definition of user names and passwords (see chapter 5). If no configuration file is provided to the server, the method returns *BadNotSupported* error code.

The method **Switch diagnostics** enables or disables diagnostic data on the server. Diagnostic data is a standard part of the OPC UA specification and is available within the *Server* node. The method may be disabled using the `ALLOW_SWITCH_DIAGNOSTICS` option in the configuration file. The diagnostics itself is controlled by the `ENABLE_DIAGNOSTICS` configuration option.

2.2 Blocks

The structure of address space within the *Exec* node reflects the structure in a target device that the server is connected to. All nodes are part of the executive name space (see chapter 4.4.2). Node names published in *BrowseName* and *DisplayName* correspond to block names in the target device.

There are two distinct objects within the *Exec* node. The first object is a so-called block. A block represents a task (TaskType), a subsystem (SubsystemType) or a function block (BlockType) on the target device. The second object is a so-called parameter. A parameter represents a single parameter of a block.

2.3 Parameters

Parameters are represented by a data type, range and actual value. A range of a data type is stored in the `Min` and `Max` nodes. A value is the only node that is constantly synchronized with the target device.

A value of a parameter is propagated immediately to the target device upon a write request from a client. However, a cached value may be returned to a client upon a read request if a value age does not exceed specified limit. A maximal age of a value is configured by a `SYNC_INTERVAL` (see chapter 4.4.1). A parameter is also refreshed

periodically within this period if a monitoring is established by a client. The process of synchronization is shown on figure 2.2.

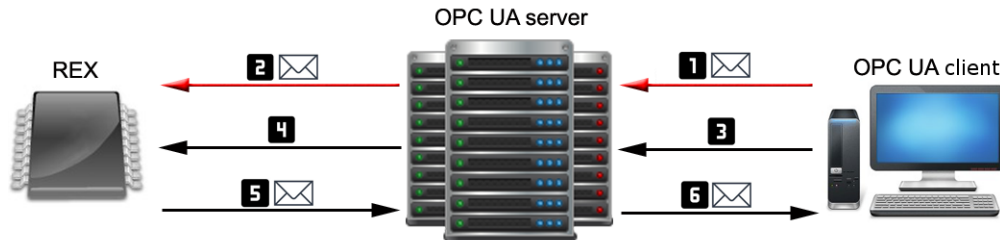


Figure 2.2: A value is stored immediately [1] to the target [2]. A value age is checked upon a read request [3]. If an age limit is exceeded, an actual value is read from the target [4] and cached internally [5]. Finally, a value is sent to the client [6].

A data type of a parameter object reflects the data type of a corresponding parameter on the target. The names published in *BrowseName* and *DisplayName* correspond to the names of corresponding parameters in the target device.

Note: Arrays and trend buffers are not supported in this version of the OPC UA server.

2.4 Events

A version number of a target executive is held internally by the sever. A *GeneralModelChangeEvent* event is invoked and list of added or removed object is passed on each time a change in the target executive is detected.

The server provides also an events for detection of communication failures. These events are inherited from *DeviceFailureEventType* and are available for the *Exec* node. An event *RexDisconnectEvent* is called if the connection to the target device is lost. An event *RexReconnectEvent* is called if a connection is established again. An event *RexReloadEvent* is called if an executive is changed on the target device.

Events					
Events Alarms Event History					
Time	Severity	Serv	Source	Message	EventType
12:33:06.017	800	RE...	Exec	REX is disconnected.	RexDisconnectEventType
12:33:13.036	750	RE...	Exec	REX is reconnected.	RexReconnectEventType

Figure 2.3: Listing of events using UaExpert

Chapter 3

Quick start guide

A configuration file, server's certificate and a private key (if encryption and authentication are required) are needed to successfully run a server. Following steps have to be done in order to start the server:

1. **Install** REX control system with the OPC UA server option enabled.
2. **Create a configuration file** by following these steps:
 - (a) Copy an example configuration from REX installation (see chapter [4.5](#)).
 - (b) Change configuration file appropriately (see chapters [4.3](#) and [6](#)).
3. **Create a certificate** if you don't have one either by using *RexOpcUaConfig* (see chapter [6.1](#)), by using *OpenSSL* or by following certificate policy of your company.
4. **Set-up user accounts** either by using *RexOpcUaConfig* or *RexOpcUaAuth* (see chapter [6](#)).
5. **Set client certificate options** if any of configured endpoint uses client certificates:
 - (a) Create certificate directories (by *RexOpcUaConfig*, see [6.1](#)).
 - (b) Copy trusted client certificates into the folder specified by the `CERTIFICATE_TRUST_LIST_PATH` configuration option.
6. **Set discovery options** appropriately if a discovery service is requested (see [4.4.6](#)):
 - (a) Find out information about your discovery server.
 - (b) Copy discovery server's certificate into corresponding folder.
 - (c) Set up configuration option in the `DISCOVERY` section.
 - i. `SERVER_URL` - Endpoint URL of a discovery server
 - ii. `SECURITY_POLICY` - Security policy used to communicate with discovery server.

- iii. `SERVER_CERTIFICATE_PATH` - A path to a certificate of a discovery server.
- iv. `ENDPOINT_URL` - Endpoint list that is to be published on a discovery server.
A single endpoint should be sufficient to register OPC UA server properly.

7. **Run the OPC UA service**, see chapter [4.2](#).

Chapter 4

Startup and configuration

4.1 Startup

The server is configured by a simple configuration file. Its location is specified by the "-c" parameter.

```
RexOpcUa [-c <configFile>]
```

Path to the configuration file is set by default in GNU/Linux:

```
/rex/OpcUa/RexOpcUa.ini
```

Configuration options are described in chapter 4. The server may also run as a system service – see chapter 4.2). A quick start guide is available in chapter 3.

4.2 System service

The OPC UA server may run as a system service. The system service mode is a default and a recommended mode.

The OPC UA server running as a service may be monitored by a *RexTrayMon* application that runs in a system tray on Windows platform. (see pictures 4.1, 4.2 a 4.3). It is also possible to start, stop and run configuration utility from *RexTrayMon*

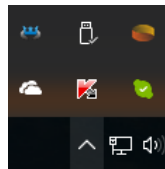


Figure 4.1: RexTrayMon application

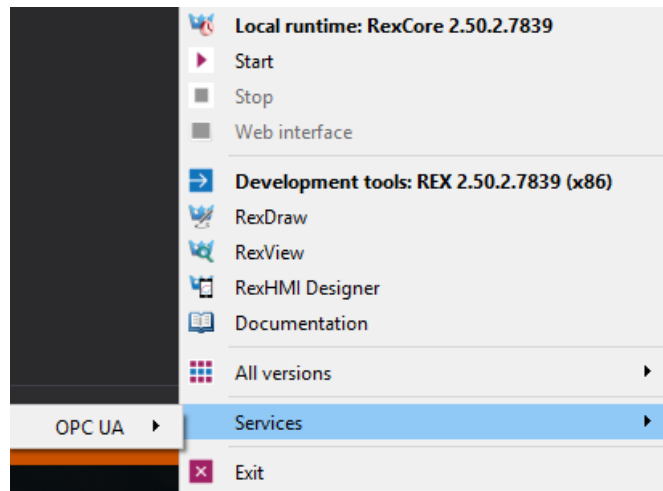


Figure 4.2: OPC UA service in RexTrayMon

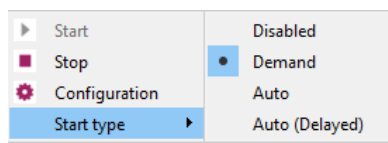


Figure 4.3: Options for OPC UA service in RexTrayMon

The OPC UA server runs as a `systemd` service on a Linux platform. The service may be started from command line using following command:

```
systemctl start rexopcu
```

Configuration file path for the server is defined by the `CFGFILE` option in a service configuration file:

```
/etc/rexcore/rexopcu.conf
```

4.3 Configuration

A configuration is stored in a standard INI file format. An UTF-8 encoding is preferred. The content is case-sensitive. There must be no additional space at the start and at the end of a line and around the "=" (equals) symbol. A comment is prefixed by a ";" (semicolon) symbol. A section name is specified within "[]" (square brackets) symbols. Every parameter must have a value. Parameters without values are ignored.

All sections must be identified by a name. A name must be recognized by the server. All recognizable section names are described in following paragraphs. Sections **Endpoint**

and **Target** may have subsections. A single corresponding endpoint or connection is created for every single subsection.

Configuration parameters are described in following paragraphs. Parameters that have a default value are optional. Parameter values may have string, number, arrays or boolean types. A number is always an integer. A boolean value is either Y, YES, ON for logical true or N, NO, OFF for logical false. An array is a set of several values within "[]" (square brackets) symbols divided by a ";" (semicolon) symbol. An empty array is considered as a no value. A file path is a system path to the file. It is either relative to the configuration file if **USE_RELATIVE_PATH** option is enabled (see table 4.2) or an absolute system path otherwise.

4.4 Configuration sections

All supported section names of a configuration file are described in following paragraphs.

4.4.1 Target

This section contains options that affect a communication link established with a target device. Details are described in the table 4.1. There is a corresponding *Exec* node build for every single *Target* identified by a name defined by **TARGET:Exec1**.

Table 4.1: Target connection settings

Field	Typ	Default value	Description
ADDRESS	IP address	–	IP address or DNS of target device that the server should connect to.
PORT	Number	–	(Optional) Communication port of the target device.
SYNC_INTERVAL	Milliseconds	500	Data synchronization period in milliseconds.
TCP_IDLE_INTERVAL	Milliseconds	30000	Idle notification period to the target in milliseconds. It should be shorter than 60000 ie. 60 seconds.
USERNAME	Text	–	(Optional) User name when authentication is required by the target.
PASSWORD	Text	–	(Optional) User password when authentication is required by the target.
USE_SSL	Y/N	N	Enables or disables secure connection using SSL.
CERTIFICATE_PATH	Optional	–	Path to a client certificate of the target device.

4.4.2 Application

The **Application** section contains main configuration options, see the table 4.2. Both the executive and server name space is configured in the **Application** section. The server name space is configured by a **APPLICATION_URI** parameter. The executive name space configuration has the following form:

```
urn:Rex:Exec:<COMPANY_NAME>:<PROJECT_NAME>:<INSTANCE_NAME>:
<TARGET_NAME>
```

Parameters **COMPANY_NAME**, **PROJECT_NAME** and **INSTANCE_NAME** should be unique for each target device. Multiple OPC UA servers that are connected to the same target device should have these parameters set to the same value. The **TARGET_NAME** parameter matches the subsection **TARGET** (see the table 4.4.1).

Should the server run on the same device on which a REX runtime core is also running, the server may be licensed as a standard component of the REX control system. Otherwise it is necessary to put a licensing key into configuration file by specifying the **REX_LICENCE_KEY**. If no valid license is present, the server terminates after one hour of operation. To get a licensing key, please contact the company and provide a device identifier ID – so-called *Site ID*. The *Site ID* is printed to the console during server startup.

Table 4.2: Application settings

Field	Typ	Default value	Description
USE_RELATIVE_PATH	Y/N	N	Determines, whether file paths are absolute or relative to the configuration gile.
APPLICATION_CERTIFICATE_PATH	File path	–	Server’s certificate file path.
APPLICATION_PRIVATE_KEY_PATH	File path	–	Server’s private key file path.
APPLICATION_PRIVATE_KEY_PASSWORD	Text	–	(Optional) Password to the certificate file.
APPLICATION_URI	Server URI	–	Server’s URI that is used as a server name space.
REX_LICENCE_KEY	Text	–	(Optional) Licensing key.
COMPANY_NAME	Text	–	Company identification. It is published in an executive name space.
PROJECT_NAME	Text	–	Project identification. It is published in an executive name space..
INSTANCE_NAME	Text	–	Server instance identification. It is published in an executive name space.

4.4.3 Security

The **Security** section contains configuration of clients certificates ie. validation options and locations. The section is relevant only when any of the configured endpoints has security options set.

The **RexOpcUaConfig** configuration tool from the REX installation may be used to create a server certificate and directories for client certificates.

Table 4.3: Security

Field	Typ	Default value	Description
CERTIFICATE_TRUST_LIST_PATH	Directory	–	Directory for client certificates that are trusted.
CERTIFICATE_REJECTED_LIST_PATH	Directory	–	Directory in which all rejected certificates by the server are stored. Rejected certificates are not stored if this options is unset.
CERTIFICATE_REVOCATION_LIST_PATH	Directory	–	(Optional) for client certificates that have been revoked.
CHECK_SELF_SIGNATURE	Y/N	N	Checking of self-signed certificates.
CHECK_CERTIFICATE_URL	Y/N	N	Certificate URL and client's URL must match if enabled.

4.4.4 Auth

The **Auth** section contains options for client authentication and authorization. The options are described in the table 4.4. Modification of user accounts and roles is described in chapters 5.2 and 6.2.

An authentication policy is specified by a client during connection handshake. No credentials are required within an anonymous connection. Otherwise a valid user name and password and/or valid and trustworthy certificate has to be supplied by a client (see table 4.3).

A list of supported authentication policies has to be defined for every endpoint by the option `USER_TOKEN_POLICY_ID`. Three anonymous policies (`ADMIN_USER_TOKEN_POLICY_ID`, `OPERATOR_USER_TOKEN_POLICY_ID`, `GUEST_USER_TOKEN_POLICY_ID`) for corresponding three different roles are supported by the server. An endpoint may support multiple anonymous policies. A client selects a required policy.

A configuration file with user roles, accounts and encrypted passwords has to be provided if authentication using name and password is enabled. An optional parameter `OPTIONAL_ENCODING_SALT` defines an encoding salt of user passwords in the configuration file.

Three policies for certificate authentication are supported by the server (`CERT_ADMIN_USER_TOKEN_POLICY_ID`, `CERT_OPERATOR_USER_TOKEN_POLICY_ID`, `CERT_GUEST_USER_TOKEN_POLICY_ID`). A client certificate is checked during connection handshake and corresponding role is assigned to the client if the certificate is valid an accepted.

Table 4.4: Authentication and Authorization

Field	Typ	Default value	Description
ADMIN_USER_TOKEN_POLICY_ID	Policy ID	–	(Optional) An anonymous policy ID for administrator role.
OPERATOR_USER_TOKEN_POLICY_ID	Policy ID	–	(Optional) An anonymous policy ID for operator role.
GUEST_USER_TOKEN_POLICY_ID	Policy ID	–	(Optional) An anonymous policy ID for guest role.
CERT_ADMIN_USER_TOKEN_POLICY_ID	Policy ID	–	(Optional) A certificate policy ID for administrator role.
CERT_OPERATOR_USER_TOKEN_POLICY_ID	Policy ID	–	(Optional) A certificate policy ID for operator role.
CERT_GUEST_USER_TOKEN_POLICY_ID	Policy ID	–	(Optional) A certificate policy ID for guest role.
CREDENTIALS_USER_TOKEN_POLICY_ID	Policy ID	–	A policy ID for authentication by user name and password. Not required if CREDENTIALS_INI_PATH is not provided.
CREDENTIALS_INI_PATH	File path	–	A configuration file with user roles, accounts. Not required if CREDENTIALS_USER_TOKEN_POLICY_ID is not provided.
OPTIONAL_ENCODING_SALT	Text	q1we58	Encoding salt for user passwords in configuration file.

4.4.5 Endpoint

The **Endpoint** section contains configuration of OPC UA endpoints that are available for clients. Each subsection in the **Endpoint** defines a single endpoint and must therefore contain all required configuration options. All configuration options are described in table 4.5.

It is recommended not to use a local IP address for an endpoint when a discovery service is required to work. The endpoint address should have following form:

opc.tcp://<IP adresa / DNS>:<port>[/<endpoint>]

Table 4.5: Endpoint settings

Field	Typ	Default value	Description
URL	URL Endpointu	–	Endpoint URL for connection using the opc.tcp. protocol.
SECURITY_POLICY	Array	–	Allowed security policies – see details int table 4.6.
USER_TOKEN_POLICY_ID	Array	–	Allowed authentication policies – see table 4.4.

Table 4.6: Security policies

Security	Description	Encryption	Algorithm
None	No	No	–
Sign_Basic128Rsa15	Yes	No	Basic128Rsa15
SignEncrypt_Basic128Rsa15	Yes	Yes	Basic128Rsa15
Sign_Basic256	Yes	No	Basic256
SignEncrypt_Basic256	Yes	Yes	Basic256

4.4.6 Discovery

The **Discovery** section configures a discovery service. The section is optional. The **ENDPOINT_URL** parameter may contain multiple endpoints that are to be discoverable by the service. However it is recommended to specify only a single endpoint. All other endpoints should be enumerable by a client through the single endpoint specified. The **ENDPOINT_URL** parameter should match at least a single endpoint on the server.

It is necessary to provide a valid URL of a discovery server, valid security options and a certificate location using **SERVER_CERTIFICATE_PATH**. A certificate should also be registered in the discovery server. Configuration parameters of a discovery service are described in table 4.7.

Table 4.7: Configuration of a discovery service

Field	Typ	Default value	Description
ENDPOINT_URL	Array	–	(Optional) URL of a discoverable endpoint.
SERVER_CERTIFICATE_PATH	File path	–	A file path to the server certificate path.
SERVER_URL	URL	–	URL of a discovery server to which the OPC UA server is registered. The URL must start with <i>opc.tcp://</i> .
SECURITY_POLICY	Policy	–	Security policy used within the connection to a discovery server. A single policy must be used. The policy must be supported by the discovery server. For more information see table 4.6.
REFRESH_TIME	Milliseconds	30000	Registration refresh interval in milliseconds.

4.4.7 Options

The **Options** section contains all other parameters that affect server's behavior. These parameters should be modified only with detailed knowledge of OPC UA specification. All parameters in this section are optional and are described in table 4.8 and 4.9. All interval values are in milliseconds.

Table 4.8: General settings

Field	Typ	Default value	Description
MIN_SAMPLING_INTERVAL	Milliseconds	600	Minimal interval of sampling nodes.
MAX_SAMPLING_INTERVAL	Milliseconds	10000	Maximal interval of sampling nodes.
MIN_PUBLISHING_INTERVAL	Milliseconds	500	Minimal interval pro publishing data to the clients.
MAX_PUBLISHING_INTERVAL	Milliseconds	600000	Maximal interval pro publishing data to the clients.
MIN_SESSION_TIMEOUT	Milliseconds	1000	Minimal client session timeout.
MAX_SESSION_TIMEOUT	Milliseconds	600000	Maximal client session timeout
MAX_PIPED_PUBLISH_REQUEST	Number	5	Maximal count of queued requests for publishing. An error code <i>TooManyPublishRequests</i> is returned if the queue exceeds the limit.
MAX_NODES_TO_ANALYZE_PER_QUERY_REQUEST	Number	100	Maximal count of analyzed nodes in a single client request
MAX_DATA_CHANGE_MONITORING_QUEUE_SIZE	Number	1000	Maximal count of queued requests of monitored items.
MAX_EVENT_MONITORING_QUEUE_SIZE	Number	1000	Maximal count of queued requests of event items.
MAX_DATA_SETS_TO_RETURN	Number	0	Maximal count of data sets to return in a single request.
ENABLE_AUDIT_EVENTS	Y/N	N	Specifies whether an event should be fired if URL of a client does not match a URL in a certificate during creation, activation and cancellation of a session and during a service call.

Table 4.9: General settings

Field	Typ	Default value	Description
ENABLE_ DIAGNOSTICS	Y/N	N	Enables diagnostic objects on the server.
ALLOW_SWITCH_ DIAGNOSTICS	Y/N	N	Enables method for enabling/disabling diagnostic objects by a client.
MIN_ DIAGNOSTICS_ UPDATE_ INTERVAL	Milliseconds	100	Minimal interval for updating of diagnostic objects.
MAX_ DIAGNOSTICS_ UPDATE_ INTERVAL	Milliseconds	86400000	Maximal interval for updating of diagnostic objects.
MAX_SESSIONS	Number	10	Maximal number of parallel sessions opened by clients, 0 for unlimited.
MAX_SESSIONS_ PER_ENDPOINT	Number	10	Maximal number of parallel sessions opened by clients on a single endpoint, 0 for unlimited.
MAX_ SUBSCRIPTIONS	Number	20	Maximal number of subscriptions created by clients, 0 for unlimited.
MAX_ SUBSCRIPTIONS_ PER_SESSION	Number	2	Maximal number of subscriptions per session, 0 for unlimited.
MAX_ SUBSCRIPTION_ LIFETIME	Number	120000	Maximal subscription lifetime.
MAX_ MONITORED_ ITEMS	Number	200	Maximal number of monitored items, 0 for unlimited.
MAX_ MONITORED_ ITEMS_PER_ SUBSCRIPTION	Number	25	Maximal number of monitored items per session, 0 for unlimited.

4.5 Configuration templates

Several configuration templates are provided to make it easier to configure a new instance of the OPC UA server. These configurations may be used as a quick start templates for arbitrary configurations. Following configuration templates are provided:

- **MINIMAL** - a minimal configuration with and unsecured endpoint and a running REX target on localhost,
- **ENDPOINTS** - a configuration with two endpoints,
- **LICENCE** - a configuration with defined licensing key,
- **SECURITY** - a configuration for secured endpoints without authentication,
- **PASS_AUTH** - a configuration for secured endpoints and authentication with user name and password,
- **CERT_AUTH** - a configuration for secured endpoints and authentication with client certificates,
- **DISCOVERY** - a configuration with registration to a discovery server,
- **FULL** - a complete configuration.

It is recommended to always modify parameters **ADDRESS**, **COMPANY_NAME**, **PROJECT_NAME** and **INSTANCE_NAME**.

Chapter 5

Authentication and authorization

5.1 Roles and users

Three roles are defined in the OPC UA server: guest, operator and administrator. A guest is allowed to browse address space and read values of parameters and block. An operator has all the permissions that a guest have and is also allowed to write values of parameters and blocks and thus affect behavior of a target algorithm. An administrator has unlimited permissions including invocation of server methods and so affect behavior of the OPC UA server itself, see chapter 2.1. Access permissions are listed in table 5.1.

Table 5.1: Permissions

Permission	Admin	Operator	Guest
Reading values	X	X	X
Writing values	X	X	
Method invocation	X		

A role that si applied for a session is determined by a security policy that is applied on an endpoint and during authentication process. A client may only apply policies that are enabled and allowed on an endpoint. Security, authentication and authorization is ensured by a configuration of policies for endpoints, see chapter 4.4.5.

A valid path to a configuration file with roles, users and passwords has to be provided when authentication using user name and password is enabled. The configuration file is loaded during the server startup or when a method *ReloadAuth* is invoked. Use **RexOpcUaAuth** or **RexOpcUaConfig** tools to modify user accounts. These tools are integral part of REX installation.

5.2 User administration

The **RexOpcUaAuth** is a command line tool that allows modification of user accounts in a configuration file. Command arguments have following syntax:

`RexOpcUaAuth <configFile> -l`

Lists all users and their roles.

`RexOpcUaAuth <configFile> -c <username> <password> <role>`

Creates a new user with a password and a role assigned. Allowed roles are admin, operator and guest.

`RexOpcUaAuth <configFile> -p <username> <password>`

Changes user password.

`RexOpcUaAuth <configFile> -a <username> <role>`

Changes user role.

`RexOpcUaAuth <configFile> -r <username> <new_username>`

Changes user name.

`RexOpcUaAuth <configFile> -d <username>`

Deletes a user.

A path to the user configuration path has to be specified using the `CREDENTIALS_INI_PATH` in the main configuration file.

Note: If a user configuration file is missing, lost or damaged, just create an empty file and define user accounts using the tool.

Chapter 6

Configuration Tool

RexOpcUaConfig is a graphical configuration tool of the OPC UA server for REX. It simplifies a process of a server configuration. It provides a certificate generation, ini file modification and administration of user accounts. Several example configurations are provided for beginners.

A whole content of the configuration ini file is shown on the *Configuration* tab, see picture [6.1](#)). A content may be modified by a user and saved. Configuration is checked for common errors before it is saved. All errors found are listed in the *Errors* tab.

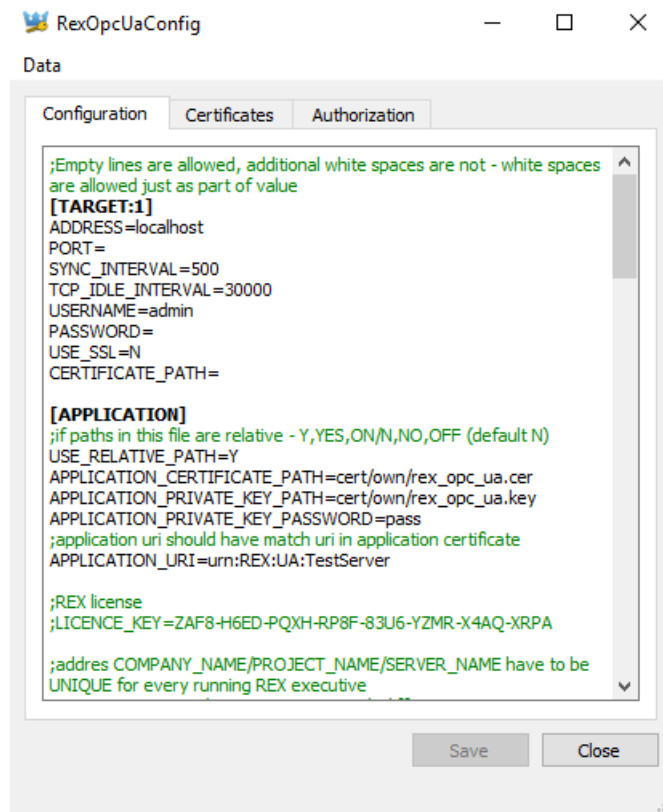


Figure 6.1: Configuration editor in RexOpcUaConfig

6.1 Certificates

Administration of server and client certificates is provided on the *Certificates* tab, see picture 6.2. All file paths are obtained from the configuration file. The configuration file must be present and all file paths must be valid, otherwise the tab is filled with a red color.

Client certificates are stored among various directories. RexOpcUaConfig makes it possible to create, open and delete these directories. Trusted client certificates are stored in the *Trusted* folder. All certificates of clients that tried to connect to the server and were rejected are stored in the *Rejected* directory.

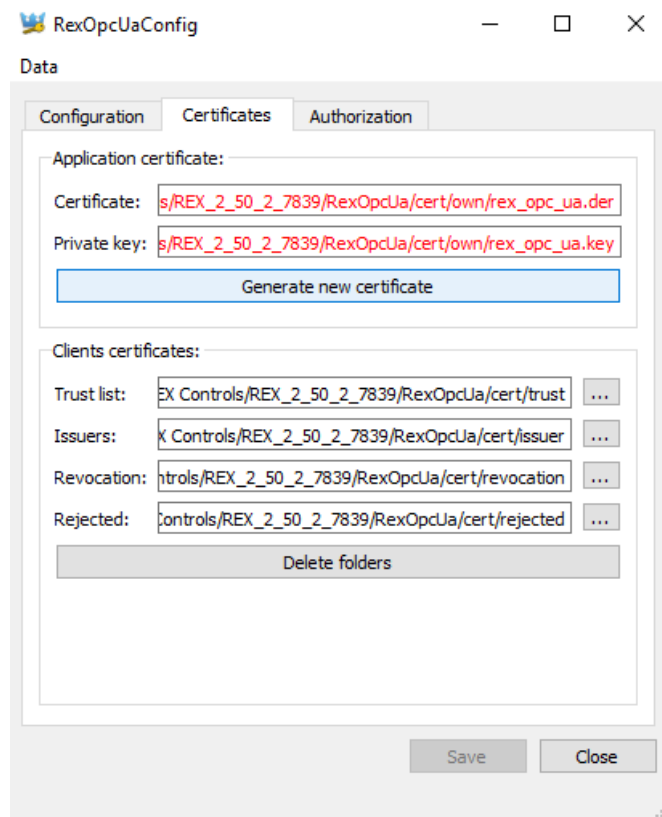


Figure 6.2: Certificate administration

A special dialog is provided for certificate creation. The *Password* and *Application URI* fields are filled by values defined by `APPLICATION_PRIVATE_KEY_PASSWORD` and `APPLICATION_URI` configuration options.

The *Subject* field contains an arbitrary text. The *Restriction* field contains an IP address or domain that the certificate is bound to. The *Application URI* must match `APPLICATION_URI` configuration option. *Certificate Settings* affect certificate's validity and used cipher.

A destination of generated file is shown on *Certificates* tab, see picture 6.2. PEM and DER file format are used for generated certificate file and private key file.

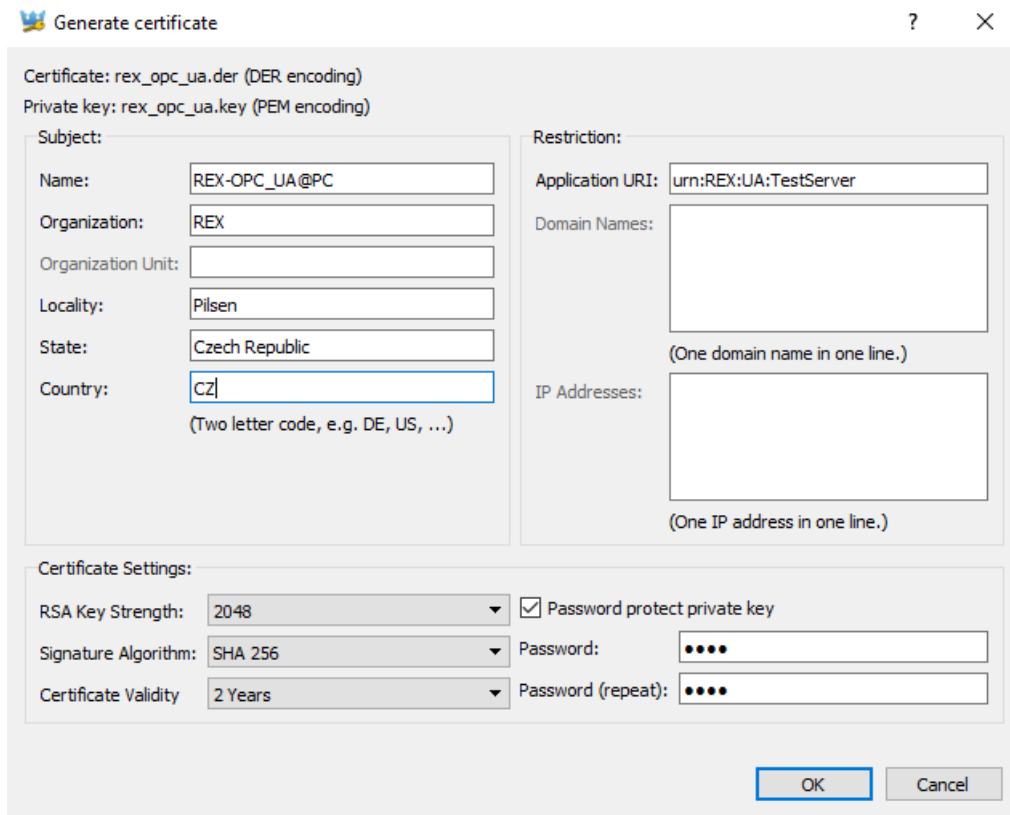


Figure 6.3: Dialog for creation of a certificate

6.2 Authentication

The *Authorization* tab (see picture 6.4) contains settings for authentication and authorization. The tab is visible only when the `CREDENTIALS_INI_PATH` is set and valid. All user accounts are stored in this configuration file.

There is a simple graphical interface for creation of a user account (picture 6.5), modification (picture 6.6) and deletion.

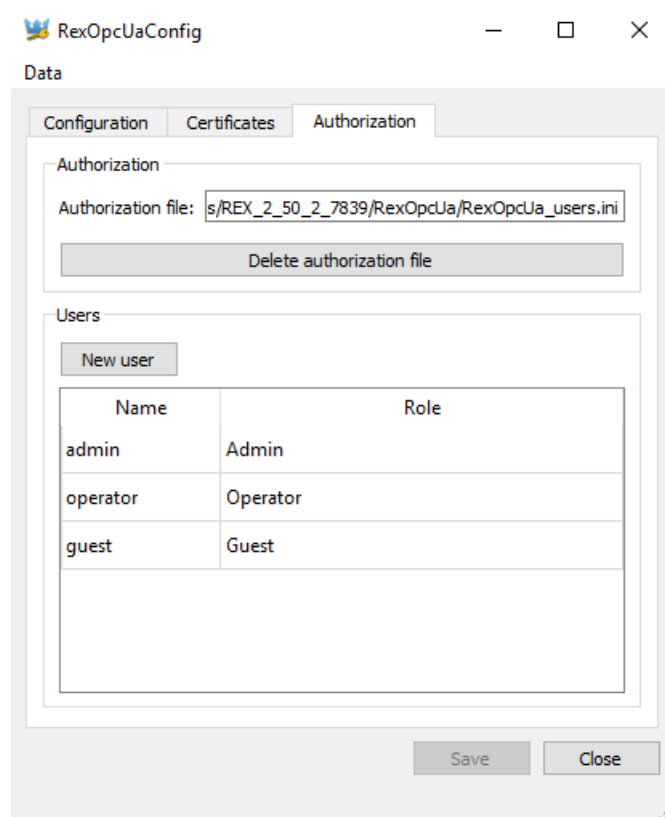


Figure 6.4: User administration

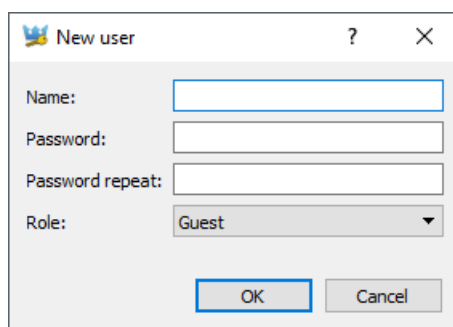


Figure 6.5: Dialog for creation of a user account

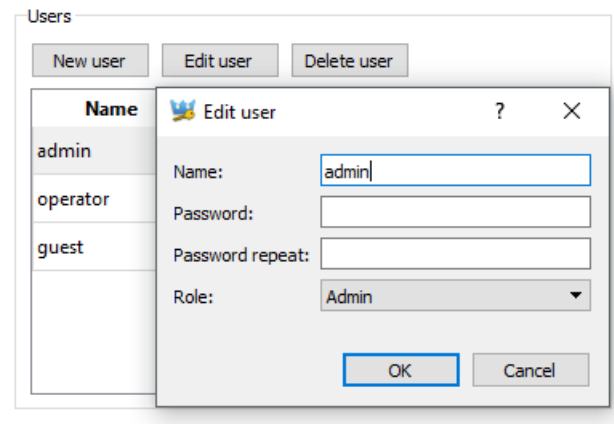


Figure 6.6: Dialog for modification of a user account

6.3 Configuration examples

Several simple configuration templates are provided for beginners. (see chapter 4.5). An example configuration may be used as a quick start template for arbitrary configurations (see pictures 6.7, 6.8 and 6.9).

No template of configuration file for user accounts exists and the file must always be created from scratch. Please check, that `OPTIONAL_ENCODING_SALT` is set appropriately when copying or re-using the configuration file.

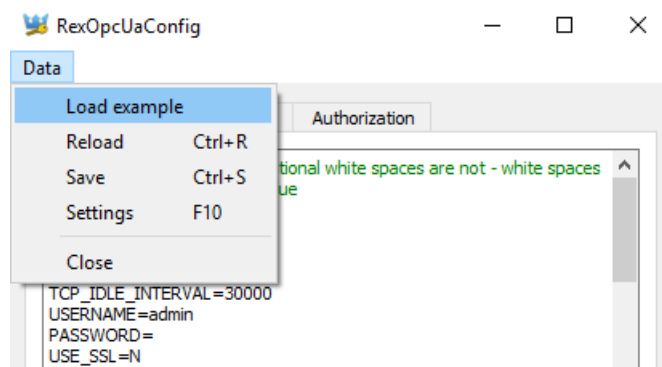


Figure 6.7: Loading of an example configuration

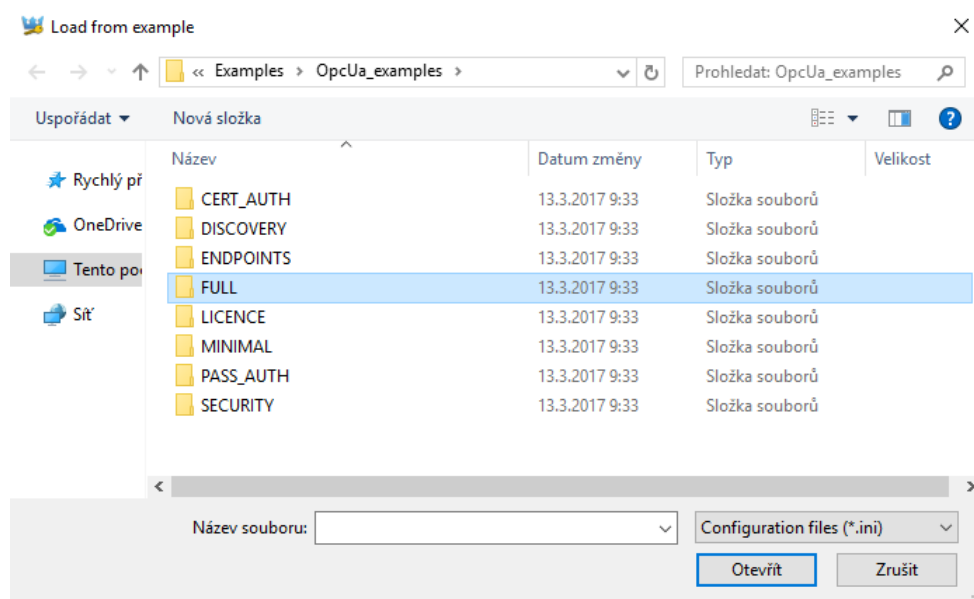


Figure 6.8: List of examples

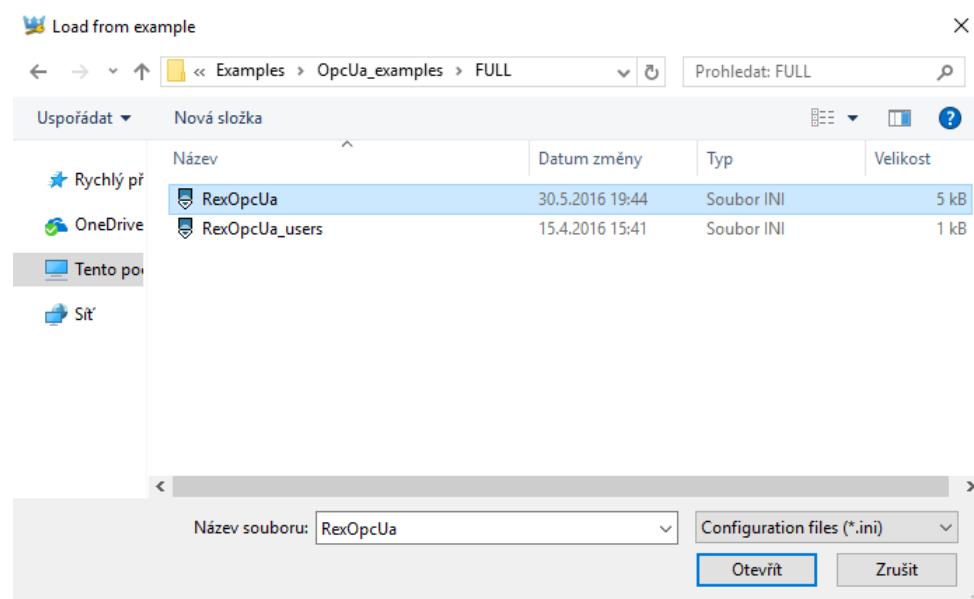


Figure 6.9: Selected configuration

Chapter 7

Connection testing with OPC UA clients

Several OPC UA clients that are freely available may be used for testing of the OPC UA server. Their behavior and functionality may differ. UaExpert by Unified Automation GmbH and myScada are shortly introduced in this guide. Both anonymous and authenticated connections are demonstrated (see pictures [7.1](#), [7.2](#) and [7.3](#)).

```
[AUTH]
;file with usernames and passwords and user token id for username/password login (optional - binded to
CREDENTIALS_INI_PATH=RexOpcUa_users.ini
CREDENTIALS_USER_TOKEN_POLICY_ID=UsernamePassword
OPTIONAL_ENCODING_SALT=q1we58
;policies for anonymous access with default privileges
ADMIN_USER_TOKEN_POLICY_ID=0
OPERATOR_USER_TOKEN_POLICY_ID=1
GUEST_USER_TOKEN_POLICY_ID=2
;policies for access with certificate
CERT_ADMIN_USER_TOKEN_POLICY_ID=AdminCertificate
CERT_OPERATOR_USER_TOKEN_POLICY_ID=OperatorCertificate
CERT_GUEST_USER_TOKEN_POLICY_ID=GuestCertificate

[ENDPOINT:1]
SECURITY_POLICY=[None,SignEncrypt_Basic256]
;policy id has to be identical to id of predefined user token policies
USER_TOKEN_POLICY_ID=[AdminCertificate,UsernamePassword,2]
URL=opc.tcp://localhost:4885/REX

[ENDPOINT:2]
SECURITY_POLICY=[None,Sign_Basic128Rsa15,SignEncrypt_Basic128Rsa15,Sign_Basic256,SignEncrypt_Basic256]
USER_TOKEN_POLICY_ID=[0]
;additional endpoint url is optional
URL=opc.tcp://localhost:4888/None/None
```

Figure 7.1: Endpoint setup without encryption


```
[AUTH]
;file with usernames and passwords and user token
CREDENTIALS_INI_PATH=RexOpcUa_users.ini
CREDENTIALS_USER_TOKEN_POLICY_ID=UsernamePassword
-----
```

Figure 7.2: Authentication with user name and password

```
[ENDPOINT:2]
SECURITY_POLICY=[None,Sign_Basic128Rsa15,SignEncrypt_Basic128Rsa15,Sign_Basic256,SignEncrypt_Basic256]
USER_TOKEN_POLICY_ID=[UsernamePassword]
;additional endpoint url is optional
URL=opc.tcp://localhost:4888/None/None
```

Figure 7.3: Endpoint policy settings

7.1 UaExpert

UaExpert is a universal and a fully functional OPC UA client that may be used for testing and verification of OPC UA connection and for a simple diagnostics. It supports wide range of OPC UA functionality.

UaExpert support three ways of authentication, encrypted connection, discovery service, reading data, writing data, monitoring of nodes, browsing address space, monitoring events, method invocation and more.

UaExpert invokes a certificate creation on a first startup. A generated certificate has to be copied to the server's trusted certificates directory if an authentication using certificate is requested using option *Settings > Manage Certificates > Copy Application Certificate To...* (see picture 7.4).

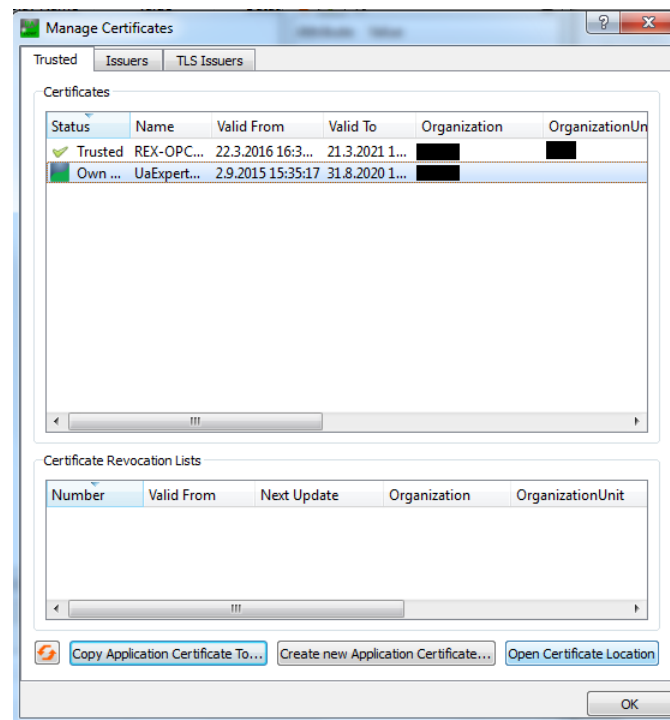


Figure 7.4: UaExpert: Storing a trusted certificate

A connection with OPC UA server is established by clicking on "+" button. A dialog for connection configuration is opened. Advanced connection options are set in tab *Advanced* (see picture 7.5).

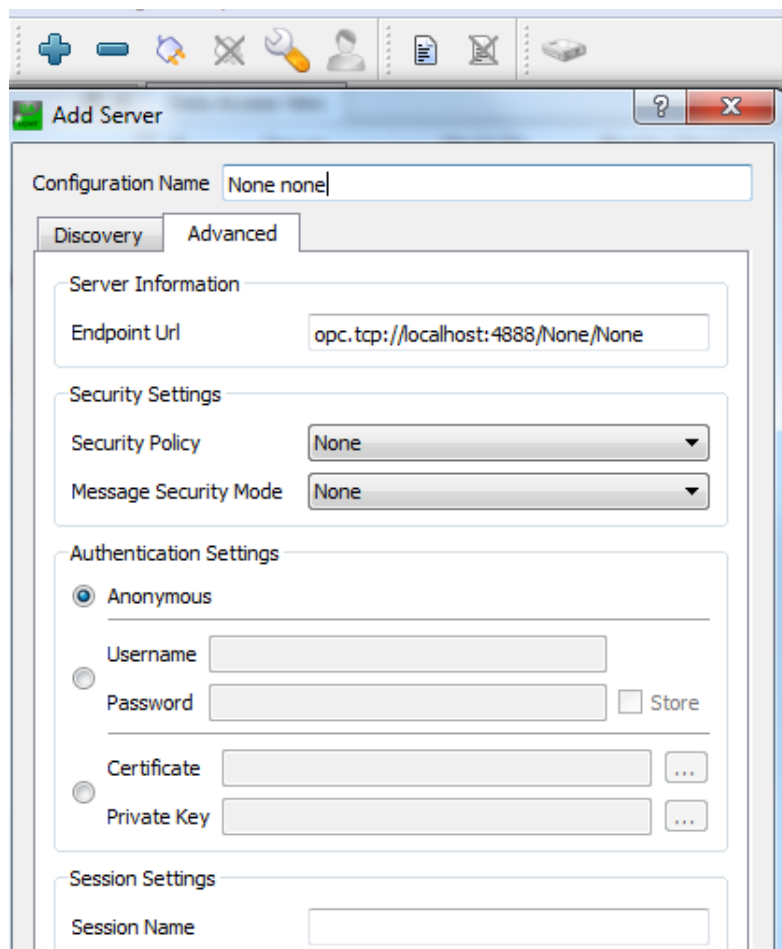


Figure 7.5: UaExpert: Anonymous connection

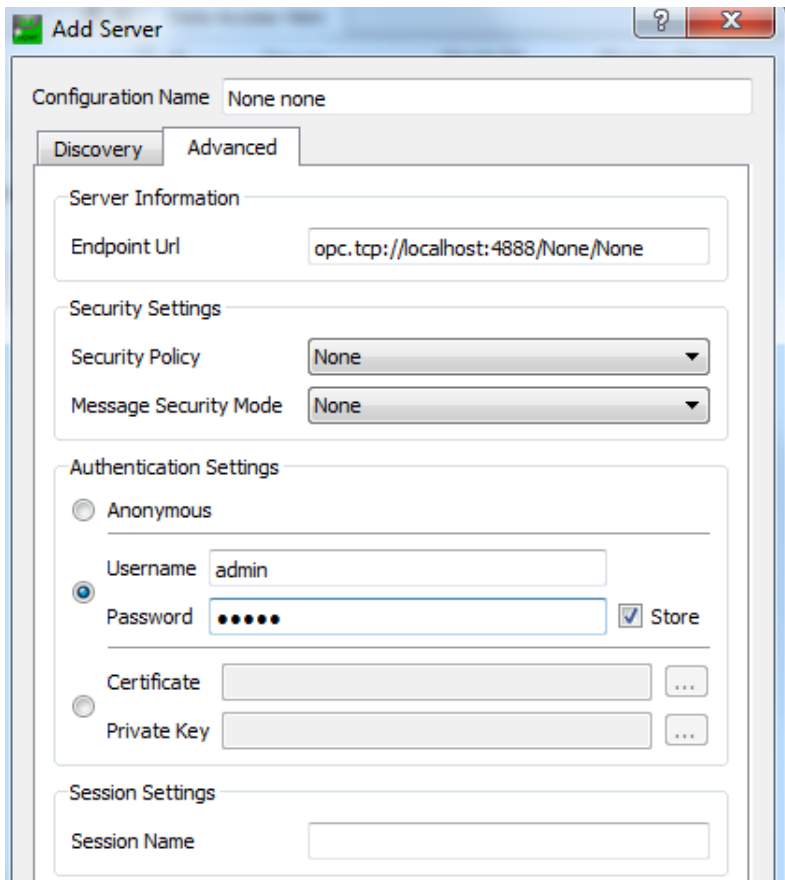


Figure 7.6: UaExpert: Connection with authentication

An established and working connection is indicated by a connected plug (see picture 7.7). A connection may be closed (unconnected plug) and re-established again. Connection options may be changed only in disconnected state (icon with a key). Authentication policy may be changed at any time (icon with a user). UaExper may have several connections established at the same time. Client configuration (including connections, monitored items etc.) may be saved and later loaded again.

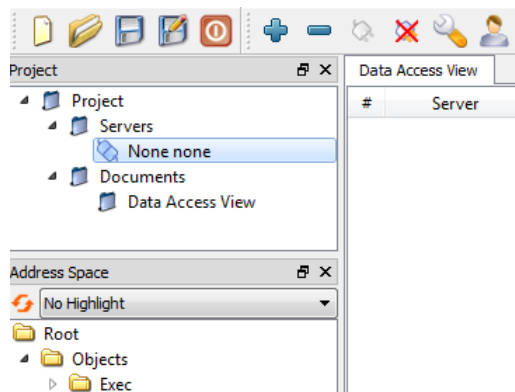


Figure 7.7: UaExpert: Connecting to the server

A *Data Access View* document has to be created by clicking on a document icon, selecting option *Data Access View* and then clicking on *Add* (see picture 7.8). To monitor an item simply drag and drop corresponding node from address space to the document (see picture 7.9). A monitoring of the item starts immediately. The item may be deleted at any time. A value may be written to a monitored item by double-clicking on the item in the document and entering new value(see picture 7.10).

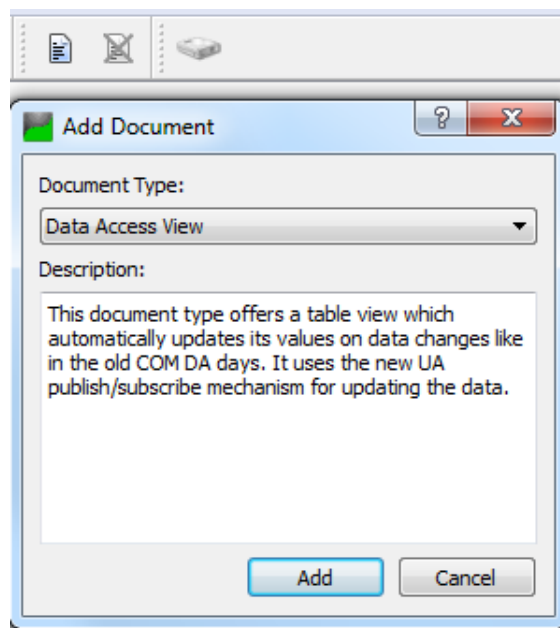


Figure 7.8: UaExpert: Selecting data for monitoring

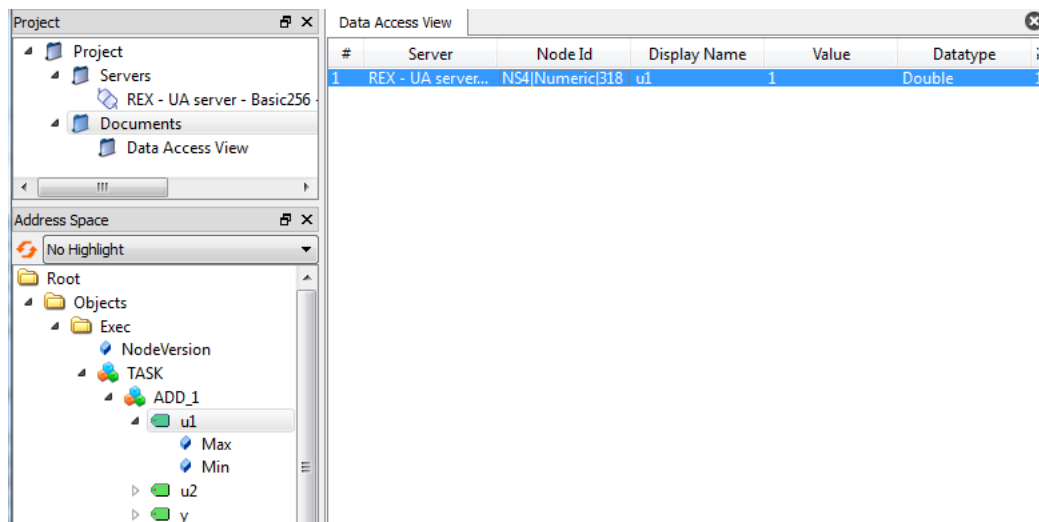


Figure 7.9: UaExpert: Monitoring of variable u1

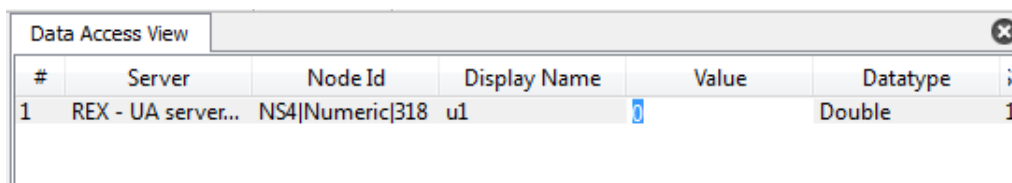


Figure 7.10: UaExpert: Writing to variable u1

A *Event View* document has to be created by clicking on a document icon, selecting option *Event View* and then clicking on *Add* (see picture 7.11). To monitor an item simply drag and drop corresponding node from address space to the *Configuration* area (see picture 7.12). All monitored events are listed in *Events* area (see picture 7.13). Event details are shown in the *Details* area. It is recommended to always monitor the *Exec* and the *Server* nodes of the OPC UA server for REX.

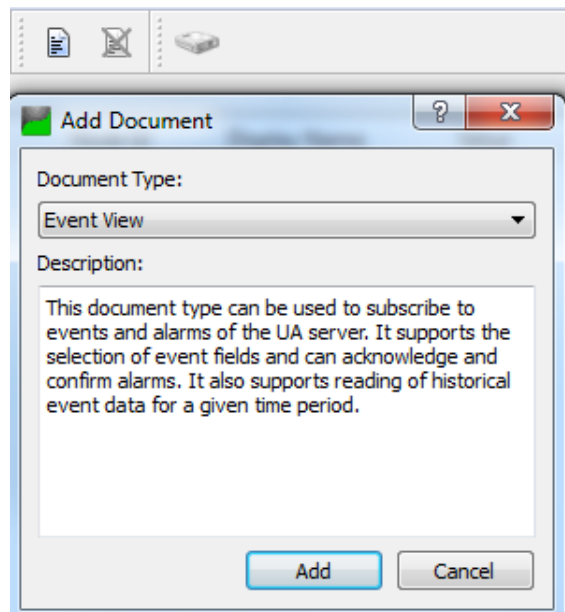


Figure 7.11: UaExpert: Adding monitored events

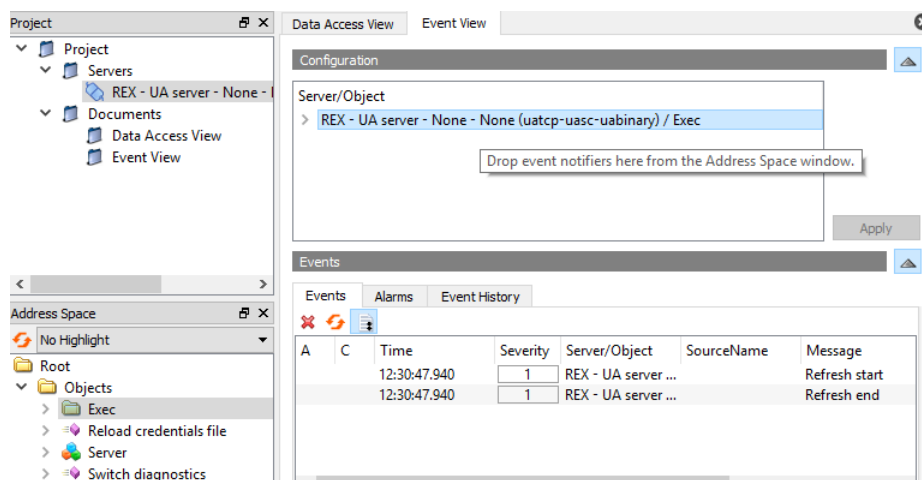


Figure 7.12: UaExpert: Monitoring events on Exec node

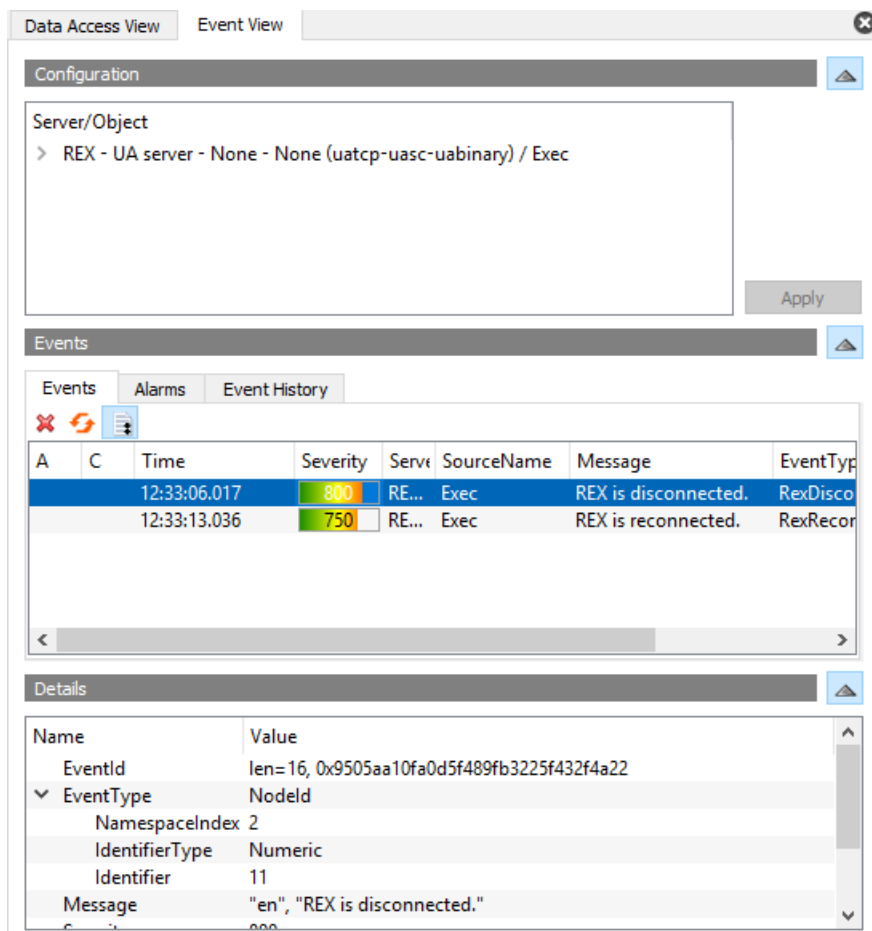


Figure 7.13: UaExpert: Listing events

A simple read operation in UaExpert is performed by clicking on a node in the address space tree view. A value is shown in the right part of the node, see picture 7.14. A write operation is invoked by double-clicking on a node, see picture 7.15.

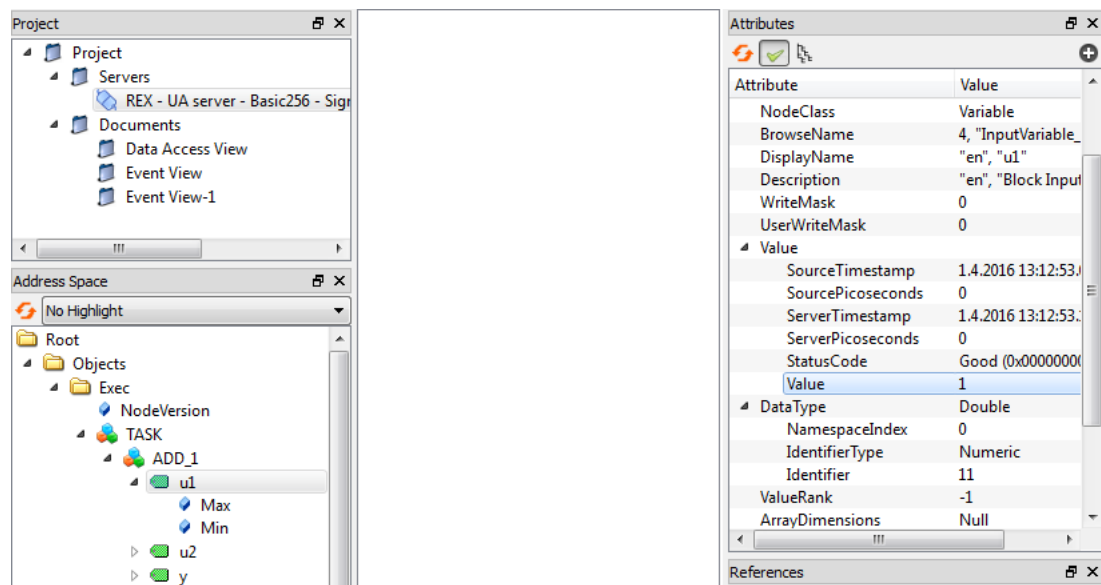


Figure 7.14: UaExpert: Reading variable u1

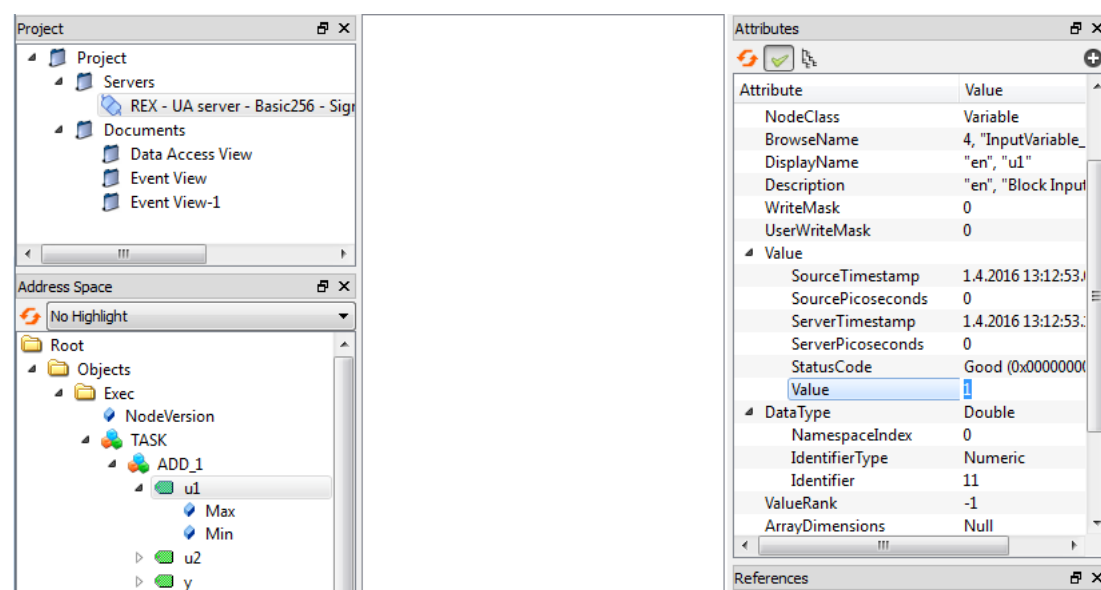


Figure 7.15: UaExpert: Writing variable u1

UaExpert supports a discovery service and shows all available OPC UA endpoints of registered servers (see picture 7.16). A user may expand a requested server in the tree view, select requested operation and set up an authentication policy to establish a

connection. UaExpert always checks a Local Discovery Server (LDS), a freely available discovery server (see chapter 4.4.6). Another option is to register a custom discovery server or OPC UA server directly.

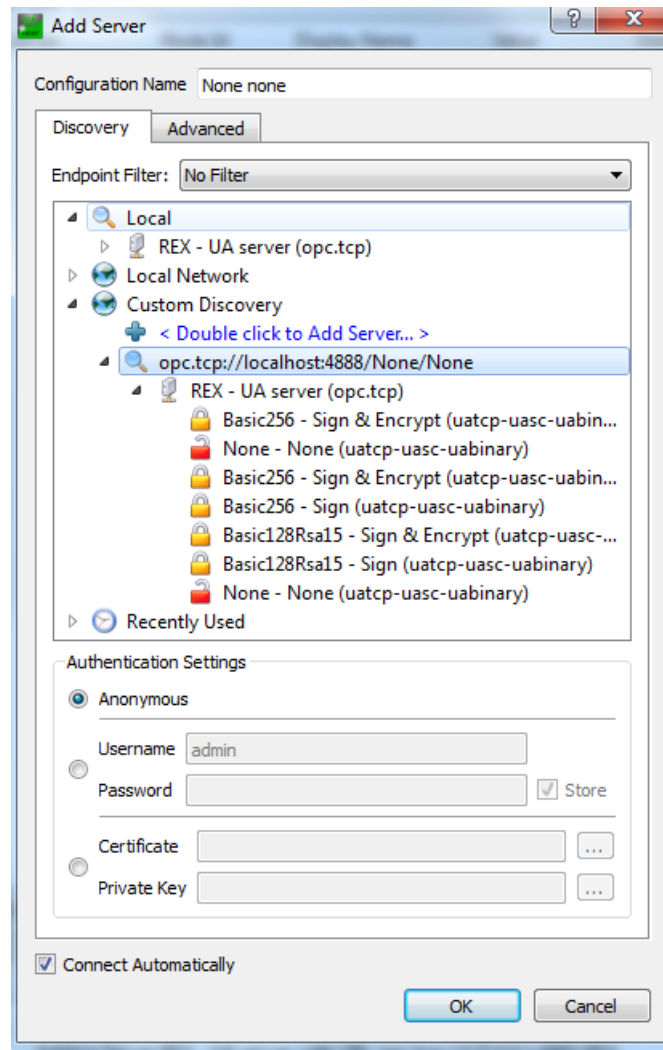


Figure 7.16: UaExpert: Using discovery service

Inspect application logs that are available in the bottom panel (see picture 7.17) if a problem occurs with a connection.



Log			
 			
Timestamp	Source	Server	Message
1.4.2016 13:20:49....	DA Plugin	REX - UA server...	Write to node 'NS4 Numeric 318' succeeded [ret = Good]
1.4.2016 13:18:56....	DA Plugin	REX - UA server...	Item [NS4 Numeric 318] succeeded : RevisedSamplingInter
1.4.2016 13:18:56....	DA Plugin	REX - UA server...	CreateMonitoredItems succeeded [ret = Good]
1.4.2016 13:18:56....	DA Plugin	REX - UA server...	Item [NS4 Numeric 318]: SamplingInterval=250, QueueSize
1.4.2016 13:18:56....	DA Plugin	REX - UA server...	Created subscription for ServerId 0

Figure 7.17: UaExpert: Logging actions

7.2 myScada

mySCADA is an Human-Machine Interface tool that supports OPC UA. Not all OPC UA options are supported by mySCADA.

First, a new project has to be created in myPROJECT designer. Then open a connection tab, insert a new connection and select OPC UA. A configuration dialog is opened in which a connection with OPC UA is set, see picture 7.18.

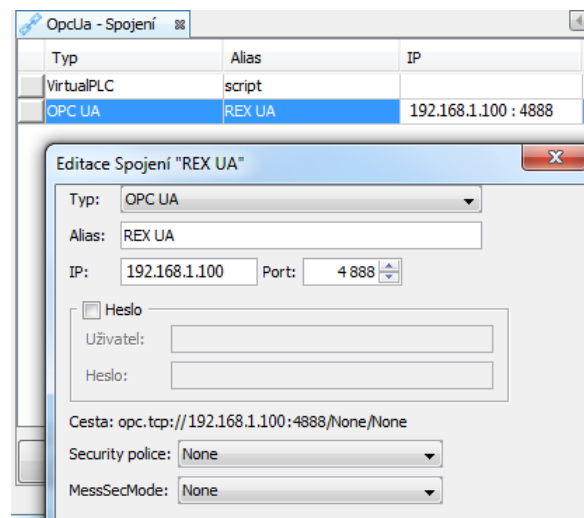


Figure 7.18: mySCADA: An anonymous connection

A data tag has to be defined in the next step. The tag contains a reference to a single node on OPC UA server that contains the requested value, see picture 7.19. The tag may be then used to show a value in HMI application, see picture 7.20. Download a final project to the device and use myView to show it.

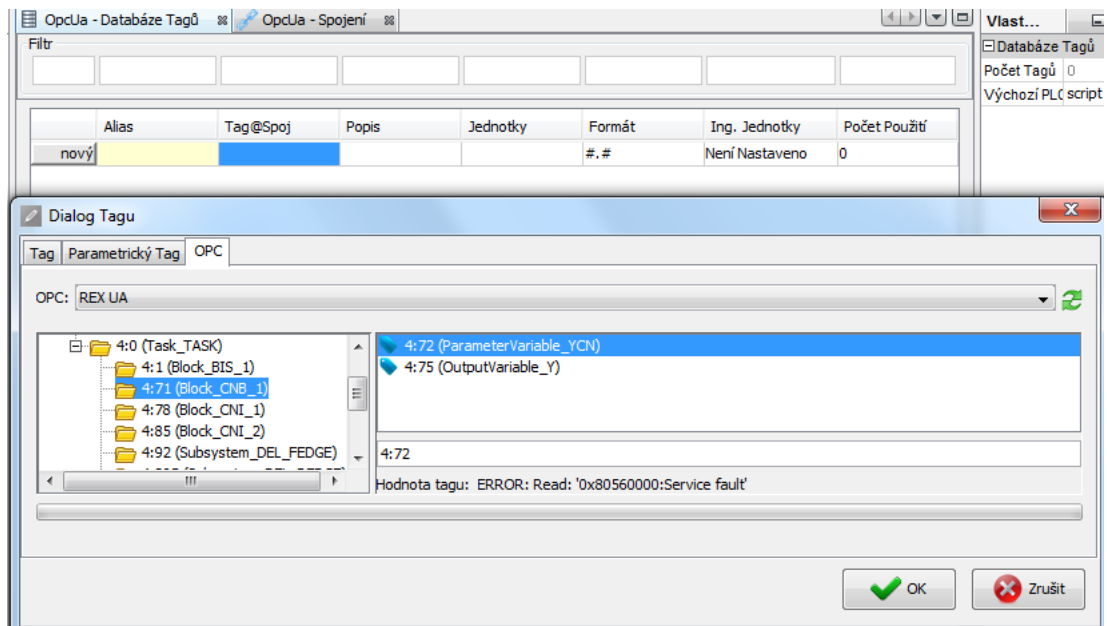


Figure 7.19: mySCADA: Tag creation

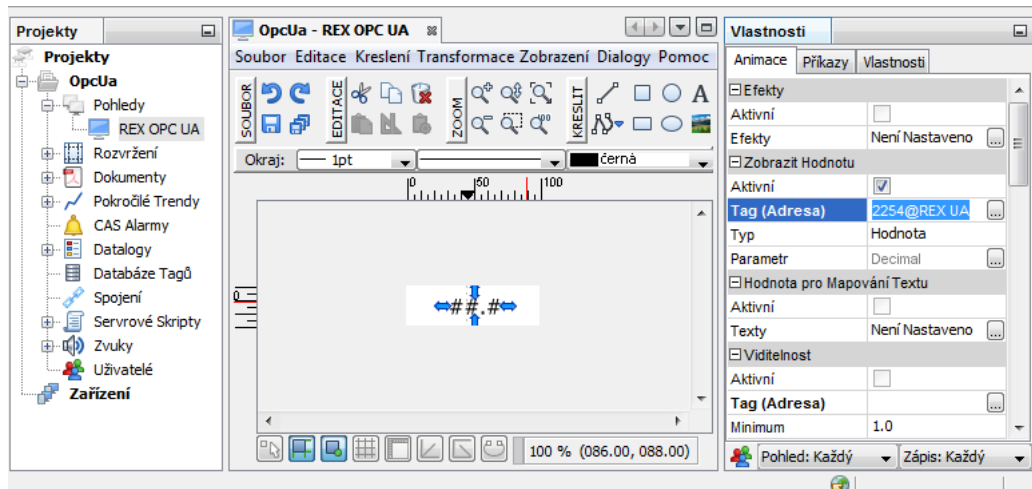


Figure 7.20: mySCADA: Using defined tag in the project

Use myView to show HMI on a device. A proper value is shown and updated from OPC UA server, see picture 7.21. An error is indicated when a connection error occurs, authentication fails or a tag is not valid, see picture 7.26.



Figure 7.21: mySCADA: Running HMI



Figure 7.22: mySCADA: Data tag is not available

An endpoint has to have a `/None/None` suffix to mySCADA work properly with unencrypted connection. A policy ID for anonymous login must be set to zero, see picture [7.1](#).

To configure user name and password authentication policy a *UserNameIdentityToken* policy has to be set (see pictures [7.23](#) and [7.24](#)) and user name and password must be supplied (see picture [7.25](#)).

```
[AUTH]
;file with usernames and passwords and user token id for
CREDENTIALS_INI_PATH= RexOpcUa_users.ini
CREDENTIALS_USER_TOKEN_POLICY_ID=UserNameIdentityToken
```

Figure 7.23: mySCADA: Authentication using user name and password

```
[ENDPOINT:2]
SECURITY_POLICY=[None,Sign_Basic128Rsa15,SignEncrypt_Basic128Rsa15,Sign_Basic256,SignEncrypt_Basic256]
USER_TOKEN_POLICY_ID=[UserNameIdentityToken]
;additional endpoint url is optional
URL=opc.tcp://localhost:4888/None/None
```

Figure 7.24: mySCADA: Endpoint policy settings

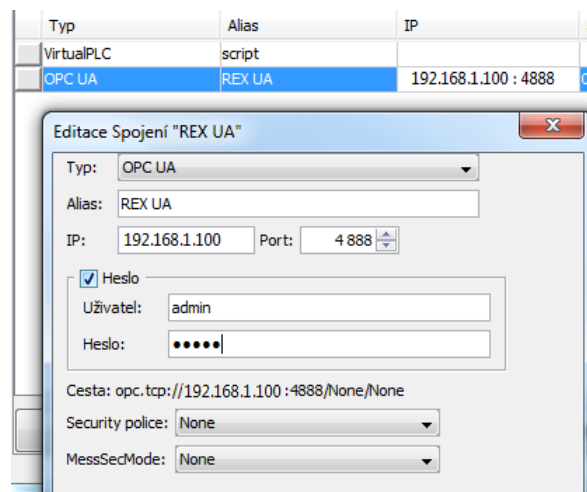


Figure 7.25: mySCADA: Authentication using user name and password

An error is indicated when authentication fails, see picture [7.26](#).

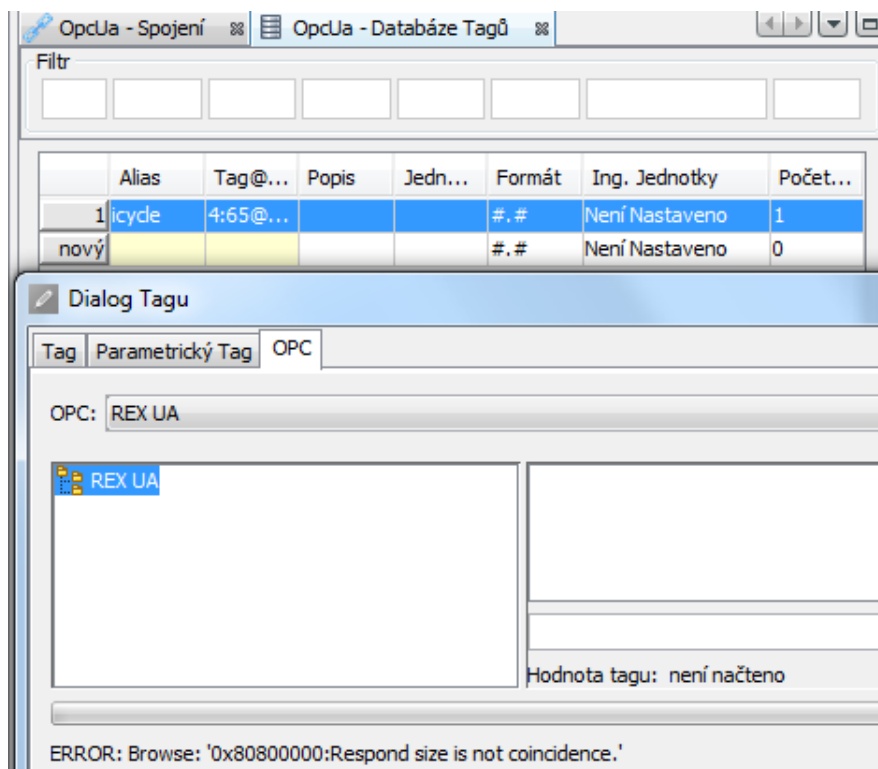


Figure 7.26: mySCADA: Connection error

Bibliography