REXYGEN www.rexygen.com

OPC UA server for REXYGEN
Reference manual

REX Controls s.r.o.

Version 3.0.4
Plzeň (Pilsen), Czech Republic
2025-03-27

Contents

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 REXYGEN

OPC UA server for REXYGEN is as standalone application that is connected to a REXYGEN runtime utilizing a low-level diagnostic protocol. It is not required to run the OPC UA server on the same station where the REXYGEN runtime is running. However, it is advised to run the OPC UA server as close to the REXYGEN runtime as possible to minimize latencies. The OPC UA server for REXYGEN 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 REXYGEN obtains the licensing information directly from the connected REXYGEN runtime. The OPC UA server itself operates unlimited. However, if an executive is too large (i.e. contains too many variables), then the OPC UA server refuses to display it. In such a case, a higher OPC UA licence has to be installed on the target REXYGEN runtime.

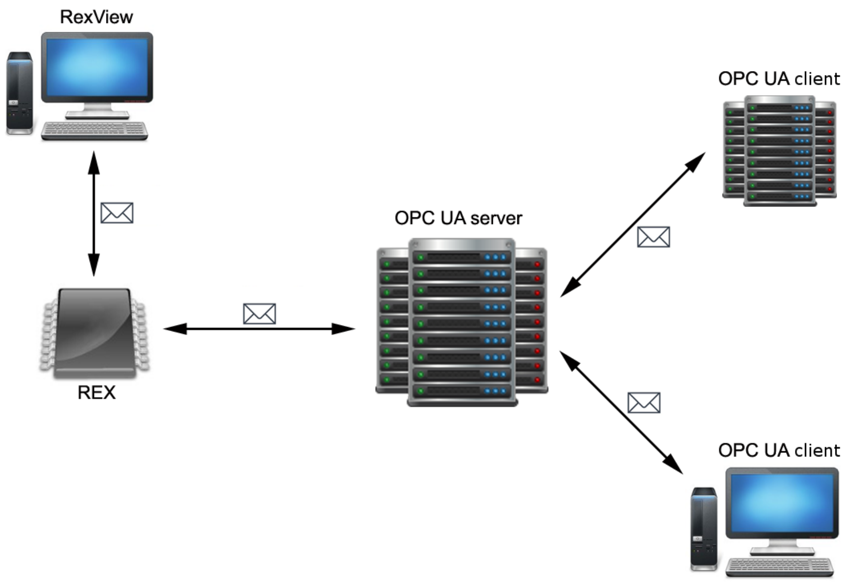


Figure 1.1: Connection between OPC UA server, OPC UA clients and REXYGEN runtime

1.3 Provided functionality

The OPC UA server is connected directly to a REXYGEN runtime core. It shows complete structure of a target algorithm (ie. blocks, variables..) 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 variables) 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).

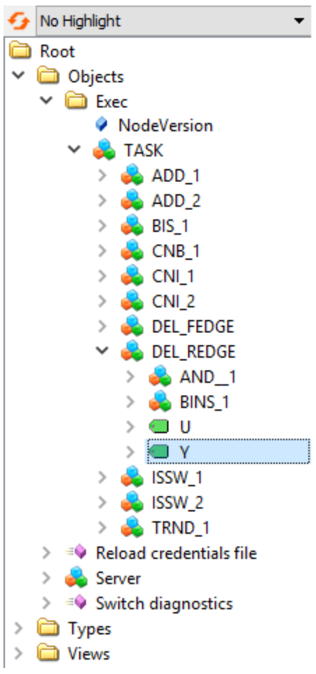


Figure 2.1: Address space in UaExpert

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 are all nodes that corresponds to the target algorithm.

2.1 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 variable. A variable represents a single input, output, state or parameter of a block.

2.2 Variables

Variables 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 variable 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 variable 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.

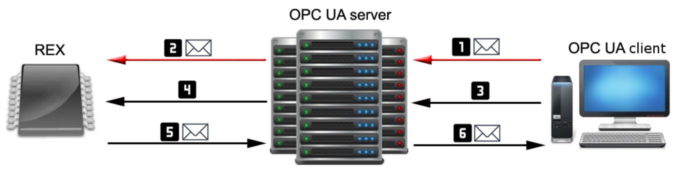


Figure 2.2: A value is stored immediately [1] to the target [2]. A value age is checked upon a read request [3]. If an age limit is exceeded, an actual value is read from the target [4] and cached internally [5]. Finally, a value is sent to the client [6].

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

Arrays (vector or matrix) are internally divided into small and large arrays. Small arrays are synchronized atomically. Large arrays are synchronized sequentially.

2.3 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.

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** REXYGEN system with the OPC UA server option enabled.
2. **Create a configuration file** by following these steps:
   (a) Copy an example configuration from REXYGEN 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 obtain it by following certificate policy of your company or generate it yourself
   (a) Using RexOpcUaConfig (viz kapitolu 6.1)
   (b) Using OpenSSL
   (c) Using script */etc/rexcore/rexopcua.d/10-cert.sh*
      - `-i <IP_ADDRESS>` – External IP address of the OPC UA server
      - `-d <DNS>` – External DNS of the OPC UA server
      - `-f` – Force regenerate certificate
      - `-k` – Force regenerate private key
4. **Set-up user accounts** either directly or by using *RexOpcUaConfig* (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 patch 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*

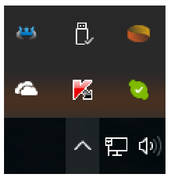


Figure 4.1: RexTrayMon application

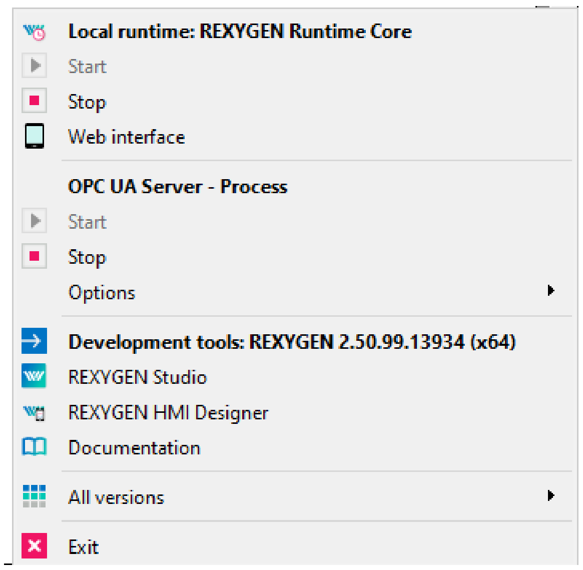


Figure 4.2: OPC UA service in RexTrayMon

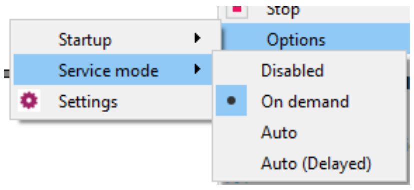


Figure 4.3: Options for OPC UA service in RexTrayMon

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

systemctl start rexopcua

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

/etc/rexcore/rexopcua.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 `User Token Policy (UTP)` `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 "," (comma) symbol. An empty array is considered as a no value. A file path is a system path to the file. It is either absolute system path or relative to the configuration file.

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

Parameter	Type	Default value	Description
CONNECTION	Target URL	–	The URL of the target device. It should follow one of these formats: `rex://[username[:password]@]host[:port]` or `rexs://[username[:password]@]host[:port]`.
SYNC_INTERVAL	Number	500	The data synchronization period, in milliseconds.
TCP_IDLE_ INTERVAL	Number	30000	The idle notification interval to the target, in milliseconds. This value should be less than 60,000 (60 seconds).
USERNAME	Text	–	(Optional) The username for authentication, if required by the target. This takes precedence over the username specified in the URL.
PASSWORD	Text	–	(Optional) The password for authentication, if required by the target. This takes precedence over the password specified in the URL.
CERTIFICATE_ PATH	Text	–	(Optional) The path to the client certificate for secure connections (`rexs` protocol).

Table 4.2: Target Device Options

Parameter	Type	Default value	Description
IGNORE_INPUTS	Y/N	N	Ignore block inputs.
IGNORE_OUTPUTS	Y/N	N	Ignore block outputs.
IGNORE_PARAMETERS	Y/N	N	Ignore block parameters.
IGNORE_STATES	Y/N	N	Ignore block states.
IGNORE_PARAMETER_ARRAYS	Y/N	N	Ignore block parameter arrays.
IGNORE_STATE_ARRAYS	Y/N	N	Ignore block state arrays.
IGNORE_LARGE_ARRAYS	Y/N	N	Ignore large block arrays.
MAX_ARRAY_SIZE	Number	65536	Maximum allowed array size [B].
SMALL_ARRAY_SIZE	Number	1024	Maximum size of small arrays [B].
LARGE_ARRAY_READ_BLOCK_SIZE	Number	1024	Block size for sequential reading [B].
LARGE_ARRAY_WRITE_BLOCK_SIZE	Number	1024	Block size for sequential writing [B].
COMMUNICATION_DIAGNOSTICS	Y/N	N	Enable communication diagnostics with REXYGENem. The CommunicationDiagnostics object will be created in the `Exec` folder.
COMMUNICATION_DIAGNOSTICS_WINDOW_WIDTH	Number	10	Interval length [s] for calculating the moving average used in diagnostics.
ADD_WHITE_LIST	Text/Array	–	Add an item to the whitelist of executive parts to be mirrored. The list is represented as a tree. Each entry corresponds to a text identifier `NodeID` to be displayed. It can be specified individually or as a list of rules.
ADD_BLACK_LIST	Text/Array	–	Add an item to the blacklist of executive parts to be ignored. The list is represented as a tree. Each entry corresponds to a text identifier `NodeID` to be ignored. It can be specified individually or as a list of rules.

Usage of Parameters ADD_WHITE_LIST and ADD_BLACK_LIST

When using `ADD_WHITE_LIST` and `ADD_BLACK_LIST`, the rules also apply to child nodes (blocks, variables) of the target node unless otherwise specified. Rules can be defined either as arrays or as individual text strings. Previously added rules are not discarded but are extended.
Example of Rule Definition:

```
ADD_WHITE_LIST=[$.task1 ,$.task2]
ADD_WHITE_LIST=[$.task3]
ADD_WHITE_LIST=$.task4
```

This setup introduces four rules, which are evaluated simultaneously.

Rules with Quantifiers

- If a rule name does not contain a colon, it applies to the node and all its child nodes (inputs, outputs, subsystems, blocks, etc.).
- Rules with a colon after the node name apply only to the inputs, outputs, parameters, states, and arrays of the node (not to its child nodes).
- Additional quantifiers can be added after the colon for filtering:
  - `I` – Node inputs
  - `O` – Node outputs
  - `P` – Node parameters
  - `S` – Node states
  - `A` – Node arrays

Example of Equivalent Rules: The following two definitions have the same meaning:

```
ADD_WHITE_LIST=$.task4:
ADD_WHITE_LIST=$.task4:IOPSA
```

Behaviour of the Blacklist

If a node is included in the blacklist, neither it nor its child nodes will be displayed in OPC UA. An exception occurs when one of its child nodes (or their inputs, outputs, parameters, states, arrays) is in the whitelist. In this case, the parent nodes will appear as "bare" to ensure the OPC UA structure remains consistent.

Note: If a node in the whitelist does not exist on the target device, its parent nodes will still appear because the rules are based on the configuration rather than the actual executive structure.

Tree Structure of Rules

The `ADD_WHITE_LIST` and `ADD_BLACK_LIST` rules form a tree structure. Each rule name is split by dots (quantifiers prefixed with a colon are not considered at this stage). If the names differ at certain levels, a new branch is created. It is also possible to use an asterisk (*) instead of a specific level, which acts as a fallback at that level.

Rule Visualisation:

For the following rule names:

```
$.task3
$.task3.sub2
$.task4.sub2
$.task4.sub3:
$.task4.*
$.*.*.block1
```

The following rule tree is created, where the underlined nodes represent tree nodes with specific rules:

```
$
+—— task3
|     +—— sub2
+—— task4
|     +—— sub2
|     +—— sub3
|     +—— *
+—— *
      +—— *
            +—— block1
```

When utilising WHITE_LIST a BLACK_LIST, the selected rule is also applied for child nodes (blocks, variables). A rule which is closest to a particular node is applied. If a node is explicitly mirrored then its ancestors (block, subsystem, task) are also created for organizational purposes. Examples of rules by priority (starting with the lowest priority):

- *$* – the entire executive
- *$.task1* – applies to task1, its variables, and its nested subsystems/blocks
- *$.task1:* – applies only to the variables of task1
- *$.task1.subsystem2* – applies to subsystem2, its variables, and its nested subsystems/blocks
- *$.task1.subsystem2:* – applies only to the variables of subsystem2
- *$.task1.subsystem2.block3* – applies to block3 and its variables
- *$.task1.subsystem2.block3:* – applies only to the variables of block3
- *$.task1.subsystem2.block3.param4* – applies to the variable param4

Searching in Rules

The following rules apply when searching the rule tree:

1. At each level, specific rules are searched for.
2. If no specific rules are found, fallback rules marked with an asterisk (*) are used.
3. If no rule is found, the algorithm moves up one level (to the left).

Example:

- For $.task3, the rules $.task3 are used - they exist and are the most specific.
- For $.task3.sub1, the rules $.task3 are used - they exist and are the most specific.
- For $.task3.sub2, the rules $.task3.sub2 are used - they exist and are the most specific.
- For $.task4, no rules exist - $.task4 has no rules, $.* has no rules, and $ has no rules.
- For $.task4.sub1, the rules $.task4.* are used - $.task4.sub1 has no rules, but $.task4.* exists and is the most specific.
- For $.task4.sub2, the rules $.task4.sub2 are used - they exist and are the most specific.
- For $.task4.sub2.block1, the rules $.task4.sub2 are used - they exist and are the most specific.
- For $.task4.sub3, the rules $.task4.sub3 are used - they exist and are the most specific.
- For $.task5, no rules exist - $.task5, $.*, and $ have no rules.
- For $.task5.sub1, no rules exist - $.task5.sub1, $.task5, $.*.sub1, $.*.*, and $ have no rules.
- For $.task5.sub1.block1, the rules $.*.*.block1 are used - $.task5, $.*.sub1, and $.*.* have no rules, but $.*.*.block1 exists and is the most specific.

Note: The rules $.*.*.block1 do not apply to $.task3.sub1.block1, as the rules $.task3 are found first. **Caution:** Exceptions to rules require explicit definition. For example, a rule $.*.sub1.* can have an exception defined as $.*.sub1.*.variable1. Similarly, a rule $.task4.sub1.* can have an exception defined as $.task4.sub1.*.variable1.

4.4.2 Application

The `Application` section contains main configuration options, see the table 4.3. 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_URI_NAME>:<PROJECT_URI_NAME>:<INSTANCE_URI_NAME>:<TARGET_
NAME>
```

Parameters `COMPANY_URI_NAME`, `PROJECT_URI_NAME` and `INSTANCE_URI_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).

Table 4.3: Application settings

Parameter	Type	Default value	Description
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.
COMPANY_URI_NAME	Text	–	Company identification. It is published in an executive name space.
PROJECT_URI_NAME	Text	–	Project identification. It is published in an executive name space..
INSTANCE_URI_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 REXYGEN installation may be used to create a server certificate and directories for client certificates.

Table 4.4: Security

Parameter	Type	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) Directory for client certificates that have been revoked.
CERTIFICATE_ISSUER_LIST_PATH	Directory	–	(Optional) Directory for certificate authorities
CERTIFICATE_REVOCATION_CHECK_OPTION	N/L/S/A	N	Check of revoked certificates. N No check L Check leaves S Not self-signed A All
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 User Token Policy (UTP)

`User Token Policy` (`UTP`) sections define allowed authentication and authorization methods. The options are described in the table 4.5. Modification of user accounts and roles is described in chapters 5 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. A certificate validation can be configured in the same way as a certificate validation for secured connection (see table 4.4).

A list of supported authentication policies has to be defined for every endpoint by the option `USER_TOKEN_POLICY`. 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 for username UTP. An optional parameter `OPTIONAL_ENCODING_SALT` defines an encoding salt of user passwords in the configuration file.

Table 4.5: User Token Policy (UTP)

Parameter	Type	Default value	Description
USER_TOKEN_ POLICY_TYPE	Anonymous, Certificate, Username	–	A type the policy.
AUTH_ROLE	Supervisor, Operator, Observer, Authorize-dUser, Anonymous	–	(Anonymous, Certificate) An assigned user role.
CREDENTIALS_ INI_PATH	File	–	(Username) A configuration file with user roles, accounts.
OPTIONAL_ ENCODING_SALT	Text	q1we58	(Username) Encoding salt for user passwords in configuration file.
CERTIFICATE_ TRUST_LIST_ PATH	Directory	–	(Certificate) Folder with trusted client's certificates.
...	...	–	(Certificate) Another certificate validation parameters from table 4.4 can be used to configure the validation.

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 4.6.

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 4.6: Endpoint settings

Parameter	Type	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 4.7.
USER_TOKEN_ POLICY (UTP)	Array	–	Allowed authentication policies – see table 4.5.
OPEN_WHEN_ ALL_BROWSED	Y/N	N	Open Endpoint only when all target are browsed. Mode for poorly implemented OPC UA clients ignoring address space changed events.

Table 4.7: Security policies and level of security (red lowest (deprecated), orange medium, green high and blue ultra high)

Security	Sign	Encrypt	Algorithm
None	No	No	–
Sign_Basic128Rsa15	Yes	No	Basic128Rsa15
SignEncrypt_Basic128Rsa15	Yes	Yes	Basic128Rsa15
Sign_Basic256	Yes	No	Basic256
SignEncrypt_Basic256	Yes	Yes	Basic256
Sign_Basic256Sha256	Yes	No	Basic256Sha256
SignEncrypt_Basic256Sha256	Yes	Yes	Basic256Sha256
Sign_Aes128Sha256RsaOaep	Yes	No	Aes128Sha256RsaOaep
SignEncrypt_ Aes128Sha256RsaOaep	Yes	Yes	Aes128Sha256RsaOaep
Sign_Aes256Sha256RsaPss	Yes	No	Aes256Sha256RsaPss
SignEncrypt_ Aes256Sha256RsaPss	Yes	Yes	Aes256Sha256RsaPss

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 4.8.

Table 4.8: Configuration of a discovery service

Parameter	Type	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 *opc.tcp://*.
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 4.7.
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 4.9 and 4.10. All interval values are in milliseconds.

Table 4.9: General settings

Parameter	Type	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 *TooManyPublishRequests* 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 4.10: General settings

Parameter	Type	Default value	Description
ENABLE_ DIAGNOSTICS	Y/N	N	Enables standard diagnostic objects on the server.
ALLOW_SWITCH_ DIAGNOSTICS	Y/N	N	Enables enabling/disabling standard 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	0	Maximal number of parallel sessions opened by clients, 0 for unlimited.
MAX_SESSIONS_ PER_ENDPOINT	Number	0	Maximal number of parallel sessions opened by clients on a single endpoint, 0 for unlimited.
MAX_ SUBSCRIPTIONS	Number	0	Maximal number of subscriptions created by clients, 0 for unlimited.
MAX_ SUBSCRIPTIONS_ PER_SESSION	Number	0	Maximal number of subscriptions per session, 0 for unlimited.
MAX_ SUBSCRIPTION_ LIFETIME	Number	120000	Maximal subscription lifetime.
MAX_ MONITORED_ ITEMS	Number	0	Maximal number of monitored items, 0 for unlimited.
MAX_ MONITORED_ ITEMS_PER_ SUBSCRIPTION	Number	0	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 REXYGEN target on localhost,
- `Secured_communication` - a configuration for secured endpoints without authentication,
- `Username_Authentication` - a configuration for secured endpoints and authentication with user name and password,
- `Certificate_Authentication` - a configuration for secured endpoints and authentication with client certificates,
- `Multi_Authentication` - a configuration for multiple authentication policies,
- `Endpoints` - a configuration with two endpoints,
- `Discovery` - a configuration with registration to a discovery server,
- `Full` - a complete configuration.

It is recommended to always modify parameters `ADDRESS`, `COMPANY_URI_NAME`, `PROJECT_URI_NAME` and `INSTANCE_URI_NAME`.

Chapter 5

Authentication and authorization

5.1 Roles and users

Five roles are defined in the OPC UA server: Anonymous, AuthorizedUser Observer, Operator and Supervisor. An AuthorizedUser is allowed to browse address space. An Observer is also allowed to read values of variables and blocks. An Operator has all the permissions as Observer and is also allowed to write values of variables and blocks and thus affect behaviour of a target algorithm. A Supervisor has unlimited permissions including utilisation of communication diagnostics and invocation of server methods. Access permissions are listed in table 5.1.

Table 5.1: Permissions

Permission	Supervisor	Operator	Observer	AuthorizedUser	Anonymous
Browse	X	X	X	X	
Reading values	X	X	X		
Writing values	X	X			
Reading permissions	X				
Communication diagnostics	X				
Method invocation	X				

A role that is 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. Use RexOpcUaConfig tools to modify user accounts. This tool is integral part of REXYGEN installation.

5.2 Credentials INI file

The credentials INI file contains information about uses, their passwords and roles. This file contains five sections corresponding to OPC UA server roles: SUPERVISOR, OPERATOR, OBSERVER, AUTHORIZED_USER, ANONYMOUS. These sections contains pairs of users with encoded passwords. Passwords are encoded by SHA1 encoding of string: `<password><username><OPTIONAL_ENCODING_SALT>`. An example credential INI file is depicted bellow.

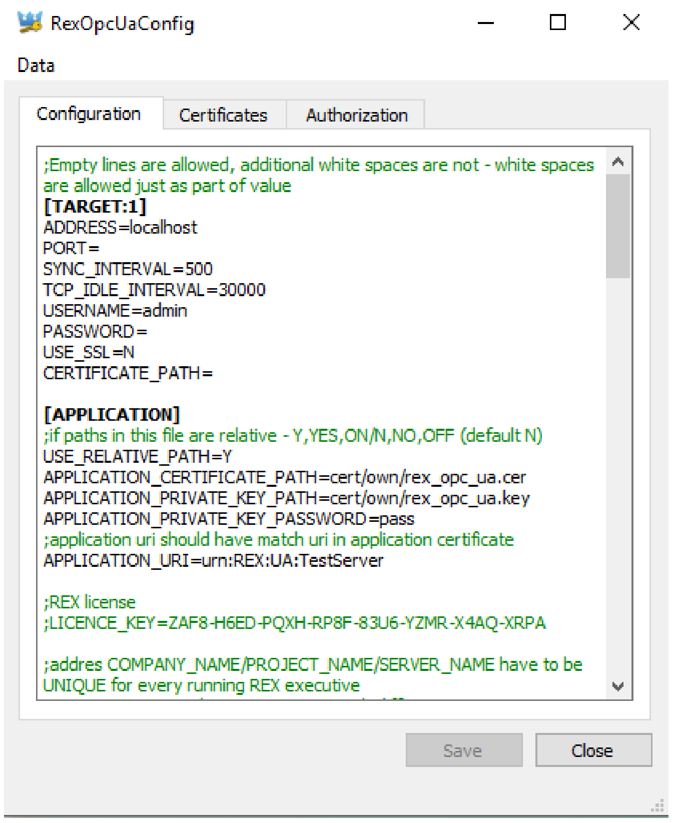
```
[SUPERVISOR]
supervisor=718DA2408623AD7786E2E79AA700E8A8FBC49221

[OPERATOR]
operator=844BD4CBFF1FEF80251306E0E359243CC267DB2B
```

Chapter 6

Configuration Tool

RexOpcUaConfig is a graphical configuration tool of the OPC UA server for REXYGEN. 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.



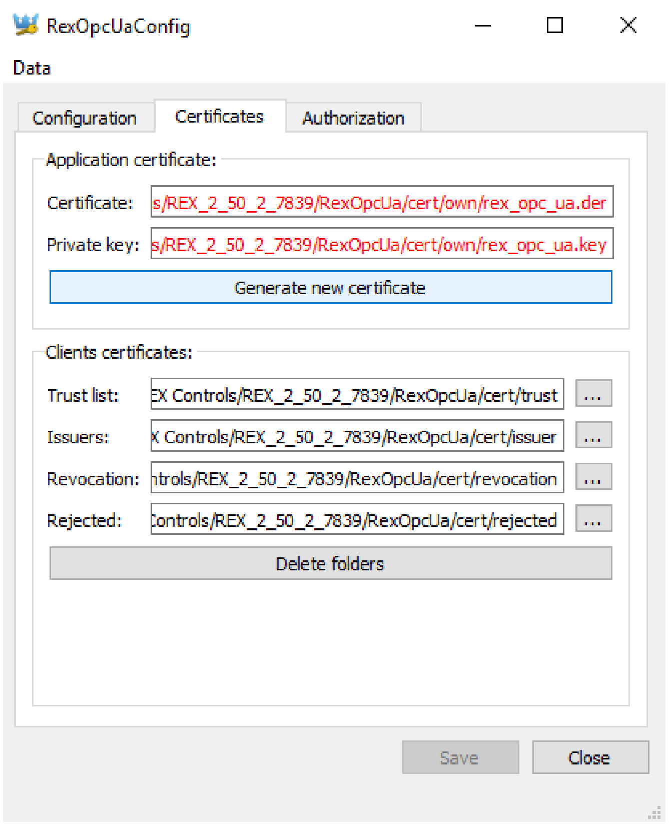
```
;Empty lines are allowed, additional white spaces are not - white spaces
are allowed just as part of value
[TARGET:1]
ADDRESS=localhost
PORT=
SYNC_INTERVAL=500
TCP_IDLE_INTERVAL=30000
USERNAME=admin
PASSWORD=
USE_SSL=N
CERTIFICATE_PATH=

[APPLICATION]
;if paths in this file are relative - Y,YES,ON/N,NO,OFF (default N)
USE_RELATIVE_PATH=Y
APPLICATION_CERTIFICATE_PATH=cert/own/rex_opc_ua.cer
APPLICATION_PRIVATE_KEY_PATH=cert/own/rex_opc_ua.key
APPLICATION_PRIVATE_KEY_PASSWORD=pass
;application uri should have match uri in application certificate
APPLICATION_URI=urn:REX:UA:TestServer

;REX license
;LICENCE_KEY=ZAF8-H6ED-PQXH-RP8F-83U6-YZMR-X4AQ-XRPA

;addres COMPANY_NAME/PROJECT_NAME/SERVER_NAME have to be
UNIQUE for every running REX executive
```

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.


RexOpcUaConfig

Data

Configuration | Certificates | Authorization

Application certificate:

Certificate: s/REX_2_50_2_7839/RexOpcUa/cert/own/rex_opc_ua.der

Private key: s/REX_2_50_2_7839/RexOpcUa/cert/own/rex_opc_ua.key

Generate new certificate

Clients certificates:

Trust list: EX Controls/REX_2_50_2_7839/RexOpcUa/cert/trust

Issuers: X Controls/REX_2_50_2_7839/RexOpcUa/cert/issuer

Revocation: ntrols/REX_2_50_2_7839/RexOpcUa/cert/revocation

Rejected: Controls/REX_2_50_2_7839/RexOpcUa/cert/rejected

Delete folders

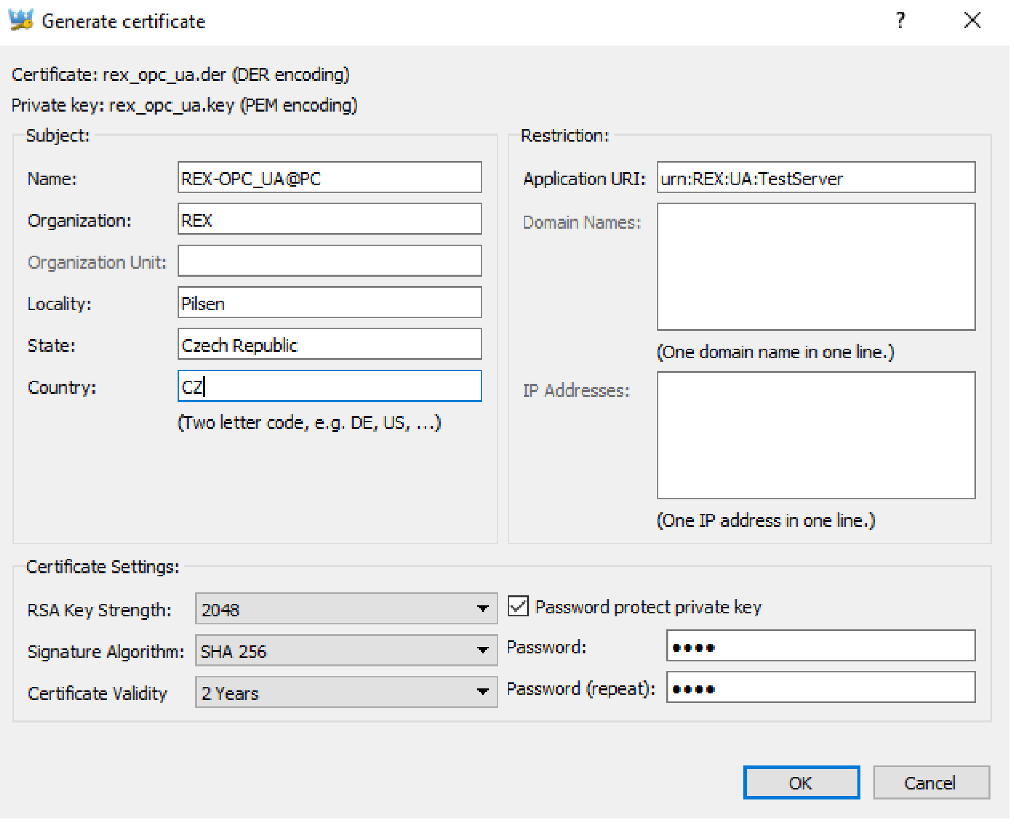
Save | Close

Figure 6.2: Certificate administration

A special dialog is provided for certificate creation. The *Passsword* and *Application URI* fields are filled by values defined by `APPLICATION_PRIVATE_KEY_PASSWORD` a `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 is shown on *Certificates* tab, see picture 6.2. `PEM` and `DER` file format are used for generated certificate file and private key file.


Generate certificate

Certificate: rex_opc_ua.der (DER encoding)
Private key: rex_opc_ua.key (PEM encoding)

Subject:

Name:	REX-OPC_UA@PC
Organization:	REX
Organization Unit:	
Locality:	Pilsen
State:	Czech Republic
Country:	CZ

(Two letter code, e.g. DE, US, ...)

Restriction:

Application URI:	urn:REX:UA:TestServer
Domain Names:	
IP Addresses:	

(One domain name in one line.)
(One IP address in one line.)

Certificate Settings:

RSA Key Strength:	2048
Signature Algorithm:	SHA 256
Certificate Validity	2 Years

☑ Password protect private key

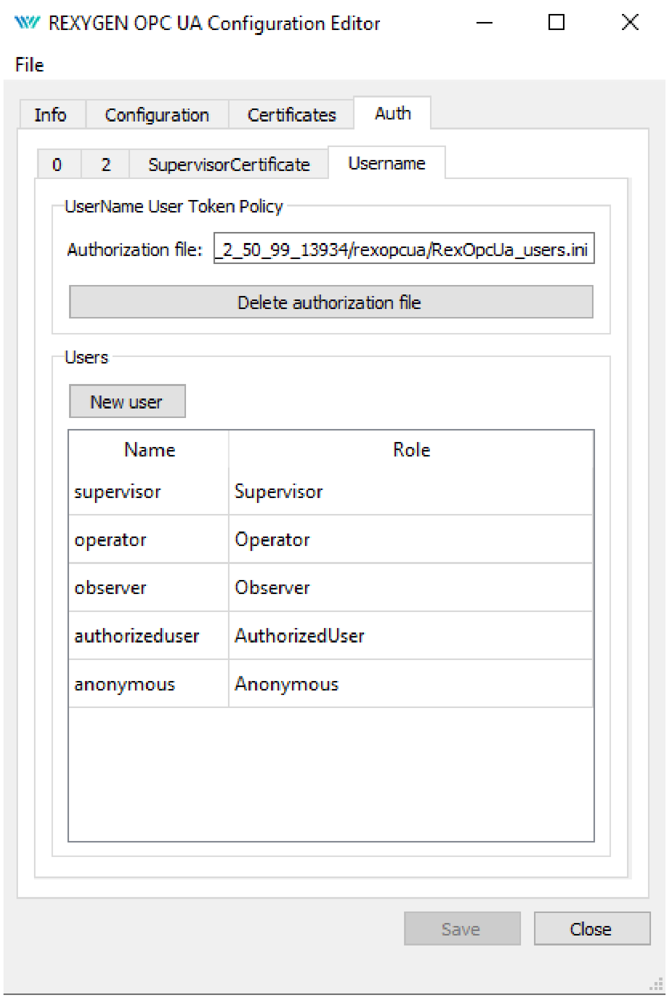
Password:	••••
Password (repeat):	••••

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.


REXYGEN OPC UA Configuration Editor

File

Info | Configuration | Certificates | Auth

0 | 2 | SupervisorCertificate | Username

UserName User Token Policy

Authorization file: _2_50_99_13934/rexopcua/RexOpcUa_users.ini

Delete authorization file

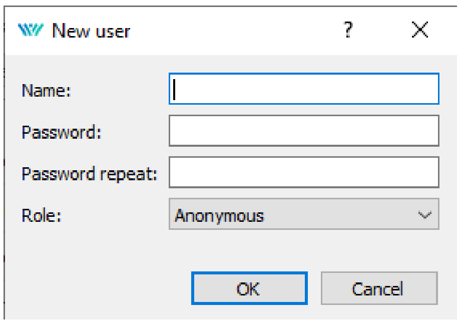
Users

New user

Name	Role
supervisor	Supervisor
operator	Operator
observer	Observer
authorizeduser	AuthorizedUser
anonymous	Anonymous

Save Close

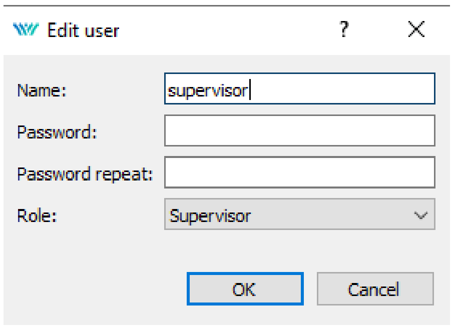
Figure 6.4: User administration


New user ?

Name:	
Password:	
Password repeat:	
Role:	Anonymous

OK Cancel

Figure 6.5: Dialog for creation of a user account



Edit user

Name: supervisor

Password:

Password repeat:

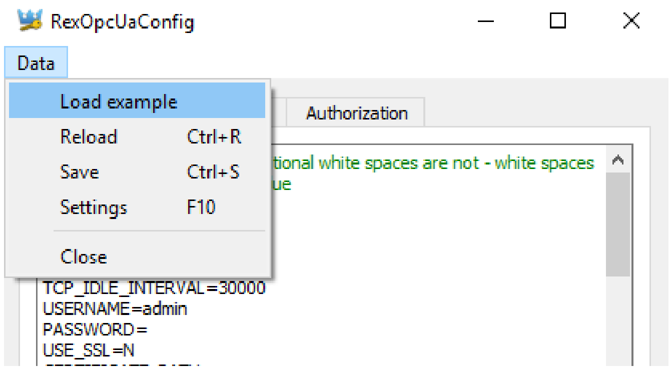
Role: Supervisor

Figure 6.6: Dialog for modification of a user account

6.3 Configuration examples

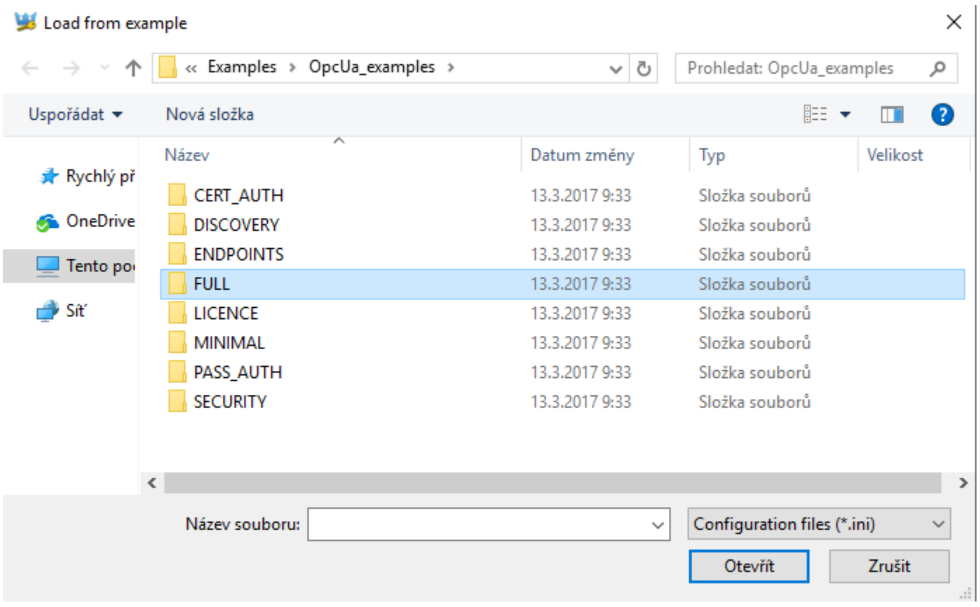
Several simple configuration templates are provided for beginners. (see chapter 4.5). An example configuration may be used as a quick start template for arbitrary configurations (see pictures 6.7, 6.8 and 6.9).

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.



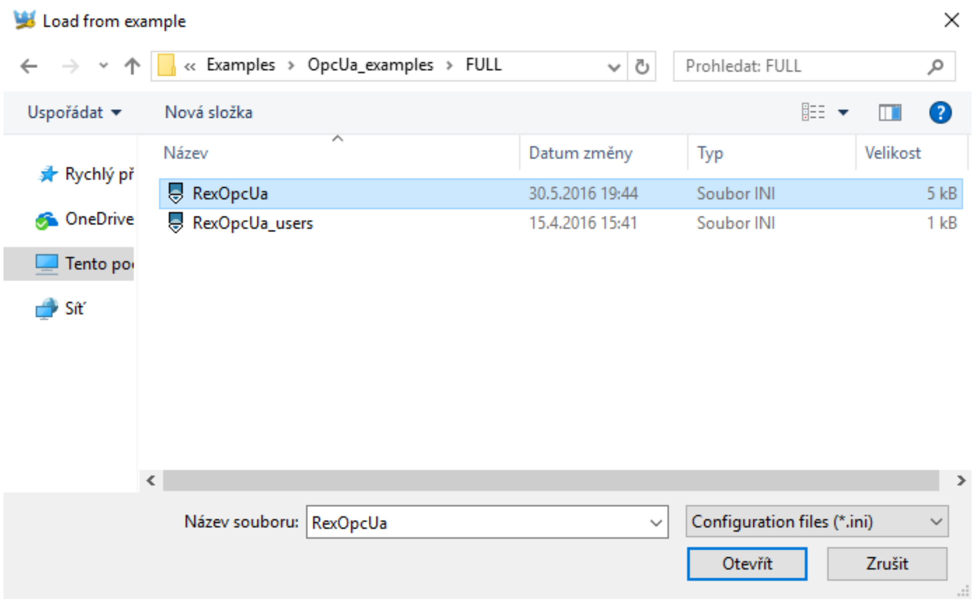
```
tional white spaces are not - white spaces
ue
TCP_IDLE_INTERVAL=30000
USERNAME=admin
PASSWORD=
USE_SSL=N
```

Figure 6.7: Loading of an example configuration



Název	Datum změny	Typ	Velikost
CERT_AUTH	13.3.2017 9:33	Složka souborů	
DISCOVERY	13.3.2017 9:33	Složka souborů	
ENDPOINTS	13.3.2017 9:33	Složka souborů	
FULL	13.3.2017 9:33	Složka souborů	
LICENCE	13.3.2017 9:33	Složka souborů	
MINIMAL	13.3.2017 9:33	Složka souborů	
PASS_AUTH	13.3.2017 9:33	Složka souborů	
SECURITY	13.3.2017 9:33	Složka souborů	

Figure 6.8: List of examples



Název	Datum změny	Typ	Velikost
RexOpcUa	30.5.2016 19:44	Soubor INI	5 kB
RexOpcUa_users	15.4.2016 15:41	Soubor INI	1 kB

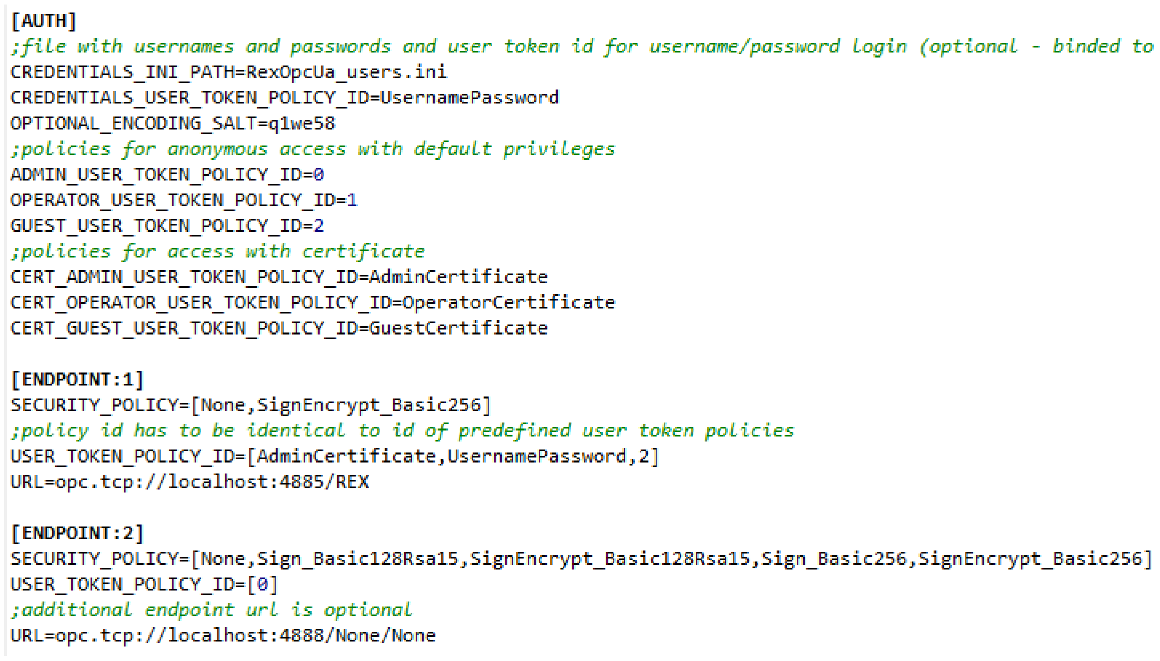
Název souboru: RexOpcUa

Figure 6.9: Selected configuration

Chapter 7

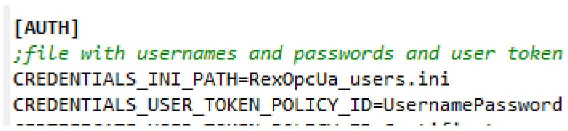
Connection testing with OPC UA clients

Several OPC UA clients that are freely available may be used for testing of the OPC UA server. Their behavior and functionality may differ. UaEpert 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).



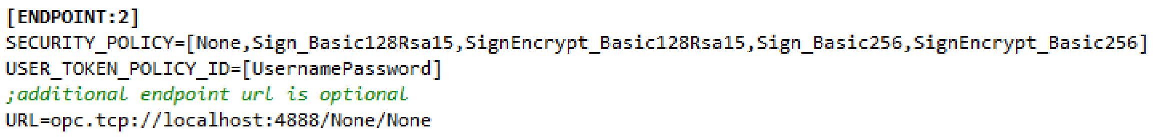
```
[AUTH]
;file with usernames and passwords and user token id for username/password login (optional - binded to
CREDENTIALS_INI_PATH=RexOpcUa_users.ini
CREDENTIALS_USER_TOKEN_POLICY_ID=UsernamePassword
OPTIONAL_ENCODING_SALT=q1we58
;policies for anonymous access with default privileges
ADMIN_USER_TOKEN_POLICY_ID=0
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
```

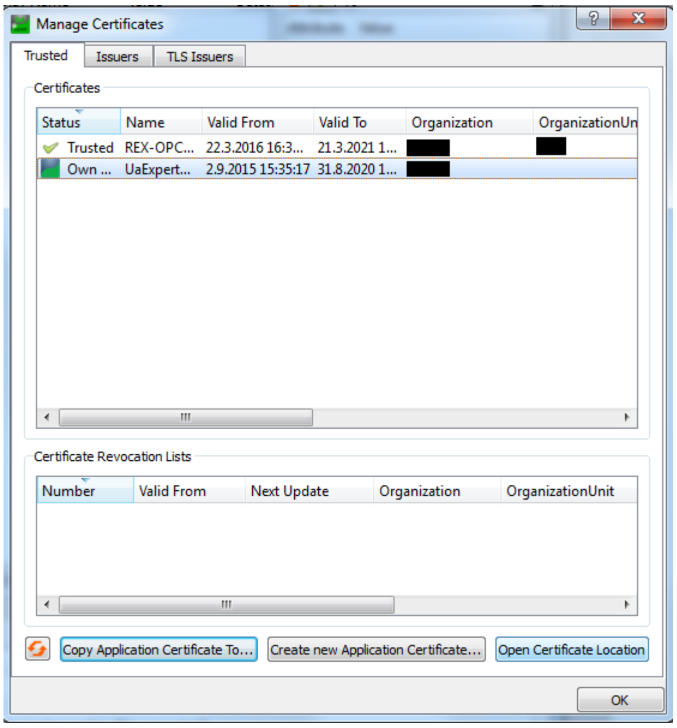
Figure 7.3: Endpoint policy settings

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).

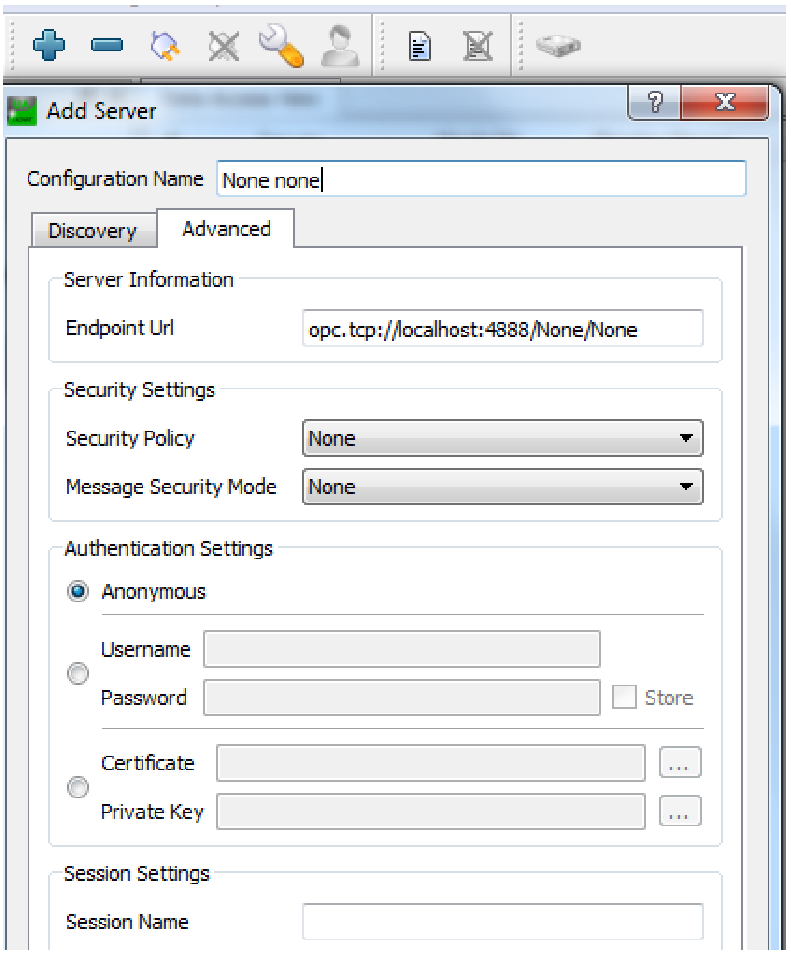

Manage Certificates
Trusted
Issuers
TLS Issuers
Certificates
Certificate Revocation Lists
Copy Application Certificate To...
Create new Application Certificate...
Open Certificate Location
OK

Status	Name	Valid From	Valid To	Organization	OrganizationUn
✓ Trusted	REX-OPC...	22.3.2016 16:3...	21.3.2021 1...	[redacted]	[redacted]
Own ...	UaExpert...	2.9.2015 15:35:17	31.8.2020 1...	[redacted]	

Number	Valid From	Next Update	Organization	OrganizationUnit

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).



Add Server

Configuration Name: None none

Discovery | Advanced

Server Information

Endpoint Url: opc.tcp://localhost:4888/None/None

Security Settings

Security Policy: None

Message Security Mode: None

Authentication Settings

(●) Anonymous

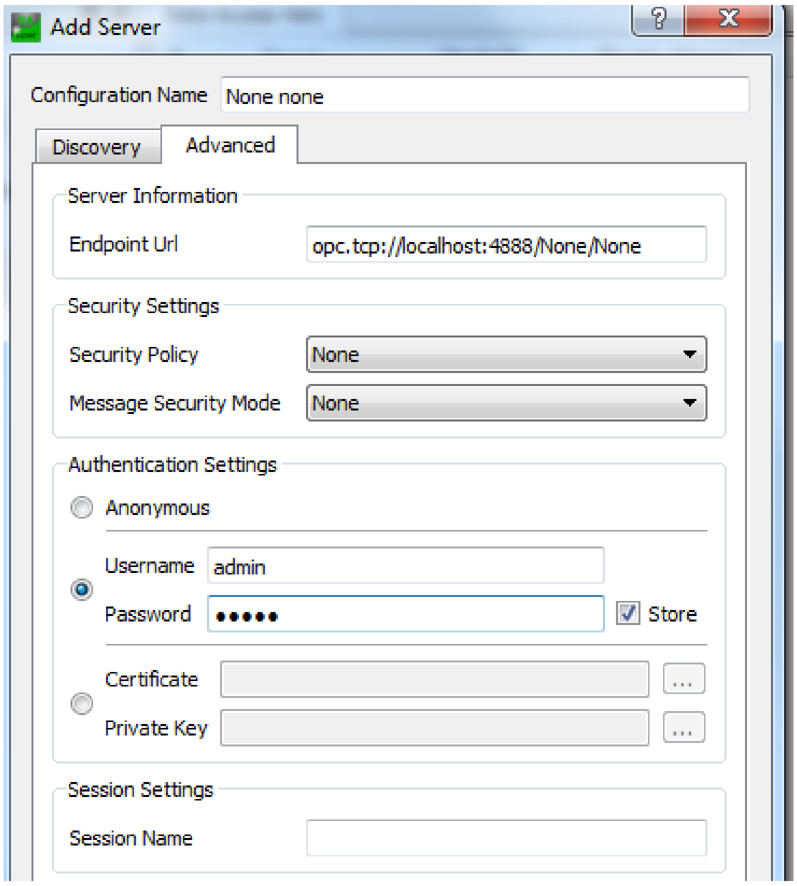
( ) Username / Password [ ] Store

( ) Certificate / Private Key

Session Settings

Session Name

Figure 7.5: UaExpert: Anonymous connection


Add Server

Configuration Name: None none

Discovery | Advanced

Server Information

Endpoint Url: opc.tcp://localhost:4888/None/None

Security Settings

Security Policy: None

Message Security Mode: None

Authentication Settings

Anonymous

Username: admin

Password: ●●●●● [✓] Store

Certificate

Private Key

Session Settings

Session Name

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.

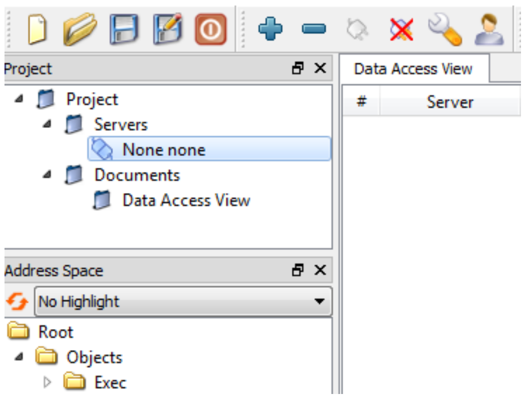
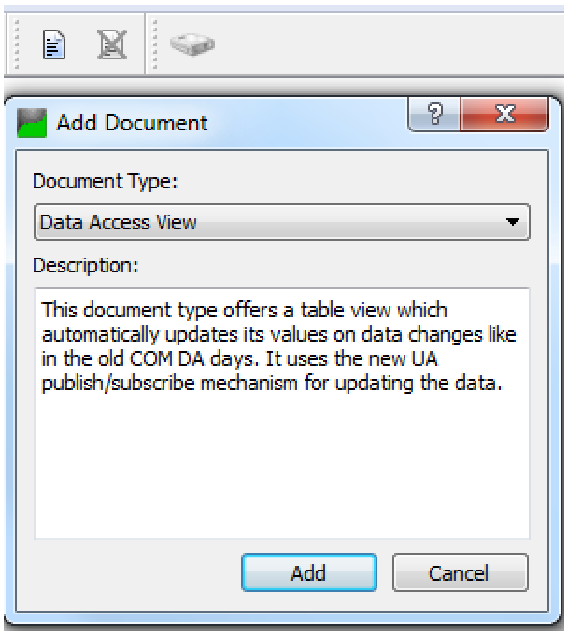


Figure 7.7: UaExpert: Connecting to the server

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).

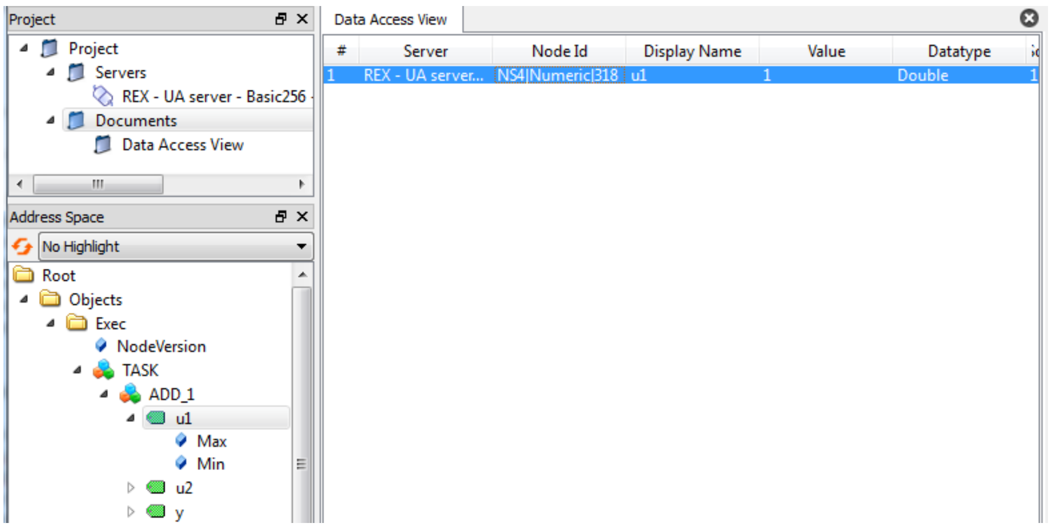


Document Type: Data Access View

Description:

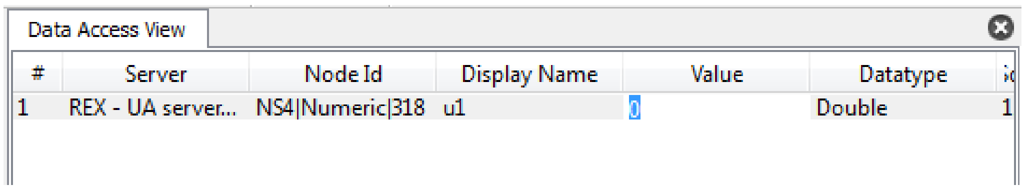
This document type offers a table view which automatically updates its values on data changes like in the old COM DA days. It uses the new UA publish/subscribe mechanism for updating the data.

Figure 7.8: UaExpert: Selecting data for monitoring



#	Server	Node Id	Display Name	Value	Datatype
1	REX - UA server...	NS4|Numeric|318	u1	1	Double

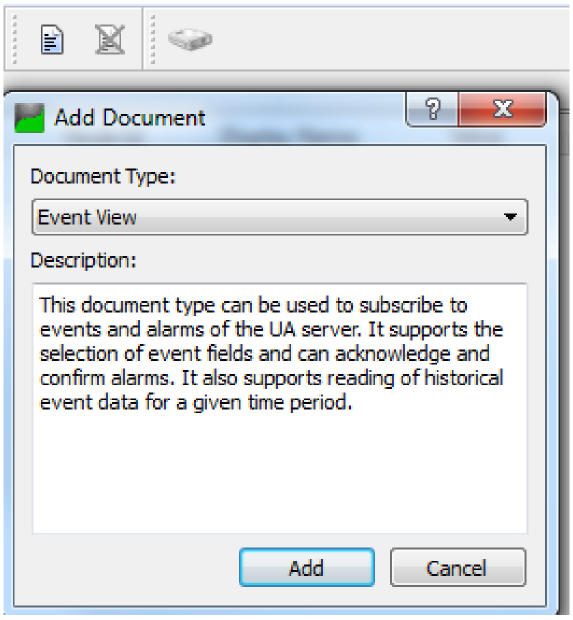
Figure 7.9: UaExpert: Monitoring of variable u1



#	Server	Node Id	Display Name	Value	Datatype
1	REX - UA server...	NS4|Numeric|318	u1	0	Double

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. 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 REXYGEN.

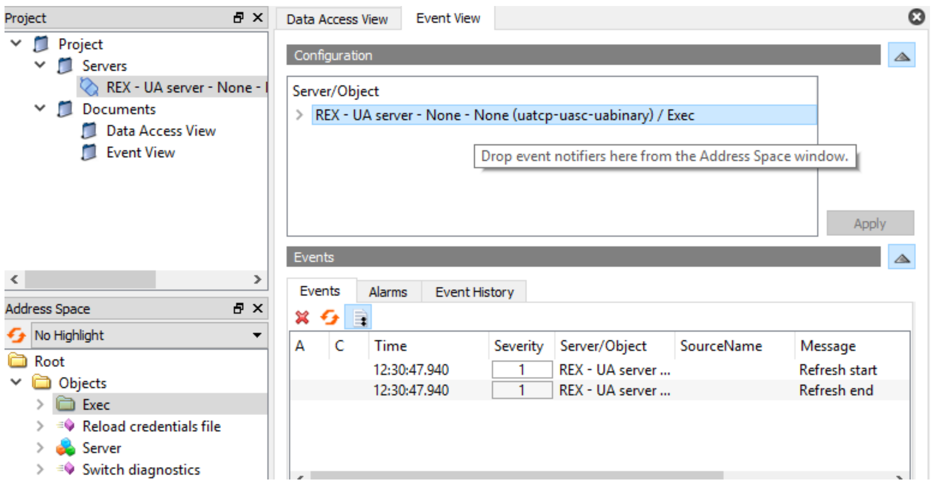

Add Document

Document Type: Event View

Description:

This document type can be used to subscribe to events and alarms of the UA server. It supports the selection of event fields and can acknowledge and confirm alarms. It also supports reading of historical event data for a given time period.

Figure 7.11: UaExpert: Adding monitored events



Server/Object

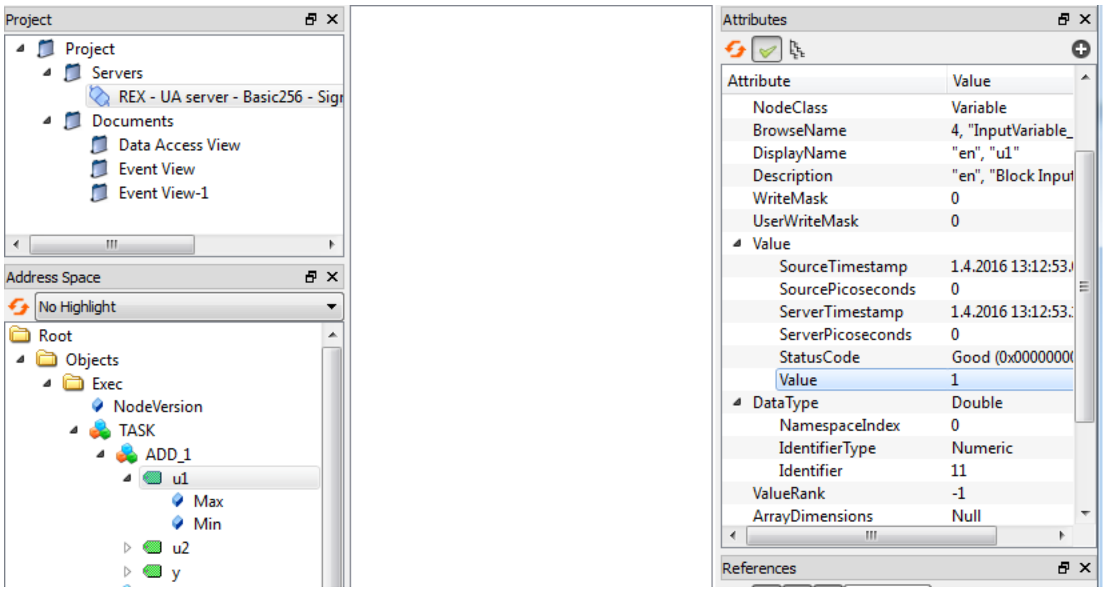
REX - UA server - None - None (uatcp-uasc-uabinary) / Exec

Drop event notifiers here from the Address Space window.

A	C	Time	Severity	Server/Object	SourceName	Message
		12:30:47.940	1	REX - UA server ...		Refresh start
		12:30:47.940	1	REX - UA server ...		Refresh end

Figure 7.12: UaExpert: Monitoring events on Exec node

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 7.13. A write operation is invoked by double-clicking on a node, see picture 7.14.


Project
Project
Servers
REX - UA server - Basic256 - Sigı
Documents
Data Access View
Event View
Event View-1

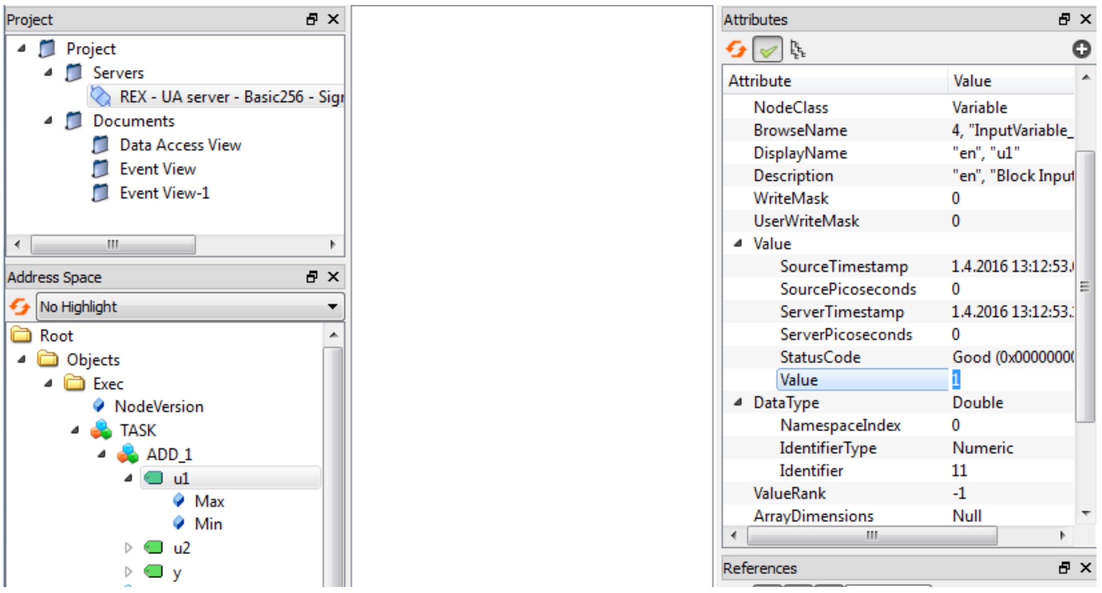
Address Space
No Highlight
Root
Objects
Exec
NodeVersion
TASK
ADD_1
u1
Max
Min
u2
y

Attributes

Attribute	Value
NodeClass	Variable
BrowseName	4, "InputVariable_
DisplayName	"en", "u1"
Description	"en", "Block Input
WriteMask	0
UserWriteMask	0
Value	
SourceTimestamp	1.4.2016 13:12:53.
SourcePicoseconds	0
ServerTimestamp	1.4.2016 13:12:53.
ServerPicoseconds	0
StatusCode	Good (0x0000000
Value	1
DataType	Double
NamespaceIndex	0
IdentifierType	Numeric
Identifier	11
ValueRank	-1
ArrayDimensions	Null

References

Figure 7.13: UaExpert: Reading variable u1


Project
Project
Servers
REX - UA server - Basic256 - Sigı
Documents
Data Access View
Event View
Event View-1

Address Space
No Highlight
Root
Objects
Exec
NodeVersion
TASK
ADD_1
u1
Max
Min
u2
y

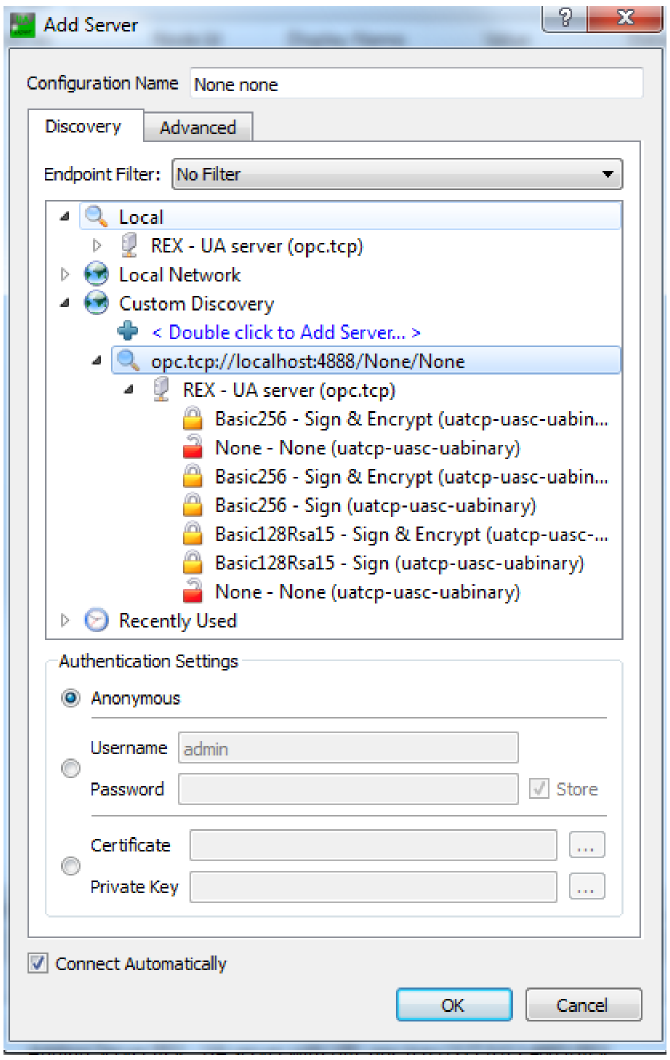
Attributes

Attribute	Value
NodeClass	Variable
BrowseName	4, "InputVariable_
DisplayName	"en", "u1"
Description	"en", "Block Input
WriteMask	0
UserWriteMask	0
Value	
SourceTimestamp	1.4.2016 13:12:53.
SourcePicoseconds	0
ServerTimestamp	1.4.2016 13:12:53.
ServerPicoseconds	0
StatusCode	Good (0x0000000
Value	1
DataType	Double
NamespaceIndex	0
IdentifierType	Numeric
Identifier	11
ValueRank	-1
ArrayDimensions	Null

References

Figure 7.14: UaExpert: Writing variable u1

UaExpert supports a discovery service and shows all available OPC UA endpoints of registered servers (see picture 7.15). A user may expand a requested server in the tree view, select requested operation and set up an authentication policy to establish a connection. UaExper 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.


Add Server

Configuration Name: None none

Discovery | Advanced

Endpoint Filter: No Filter

- Local
  - REX - UA server (opc.tcp)
- Local Network
- Custom Discovery
  - < Double click to Add Server... >
  - opc.tcp://localhost:4888/None/None
    - REX - UA server (opc.tcp)
      - Basic256 - Sign & Encrypt (uatcp-uasc-uabin...
      - None - None (uatcp-uasc-uabinary)
      - Basic256 - Sign & Encrypt (uatcp-uasc-uabin...
      - Basic256 - Sign (uatcp-uasc-uabinary)
      - Basic128Rsa15 - Sign & Encrypt (uatcp-uasc-...
      - Basic128Rsa15 - Sign (uatcp-uasc-uabinary)
      - None - None (uatcp-uasc-uabinary)
- Recently Used

Authentication Settings

[✓ Anonymous]

Username: admin
Password: Store ✓

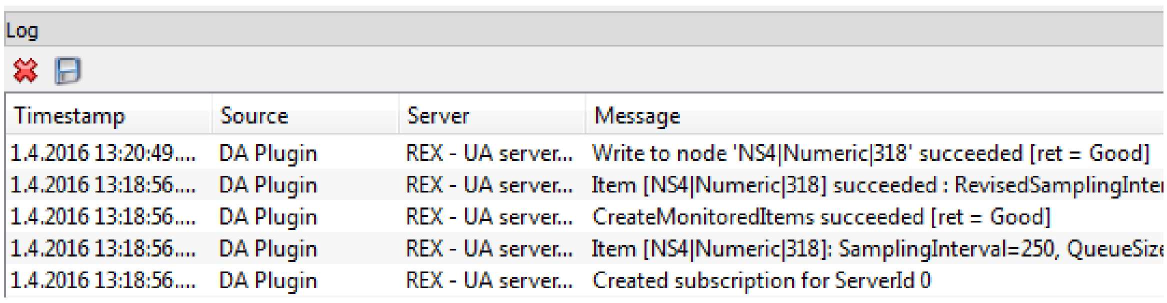
Certificate:
Private Key:

✓ Connect Automatically

OK | Cancel

Figure 7.15: UaExpert: Using discovery service

Inspect application logs that are available in the bottom panel (see picture 7.16) if a problem occurs with a connection.



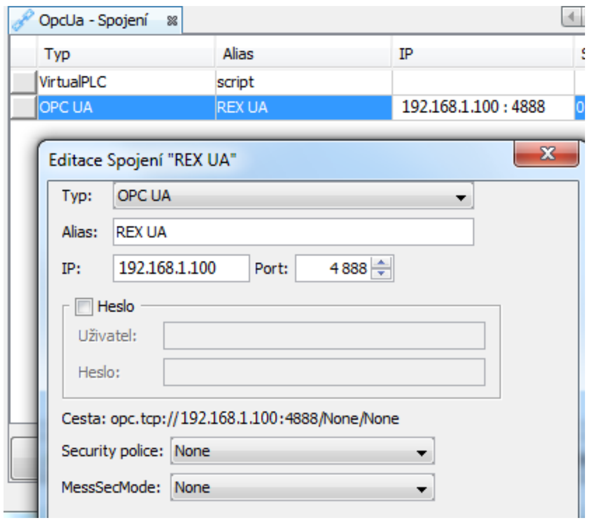
```
Log
Timestamp            Source      Server            Message
1.4.2016 13:20:49....  DA Plugin   REX - UA server...  Write to node 'NS4|Numeric|318' succeeded [ret = Good]
1.4.2016 13:18:56....  DA Plugin   REX - UA server...  Item [NS4|Numeric|318] succeeded : RevisedSamplingInte
1.4.2016 13:18:56....  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 7.16: 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 a select OPC UA. A configuration dialog is opened in which a connection with OPC UA is set, see picture 7.17.



OpcUa - Spojení

Typ	Alias	IP
VirtualPLC	script	
OPC UA	REX UA	192.168.1.100 : 4888

Editace Spojení "REX UA"

Typ: OPC UA

Alias: REX UA

IP: 192.168.1.100 Port: 4 888

Heslo

Uživatel:

Heslo:

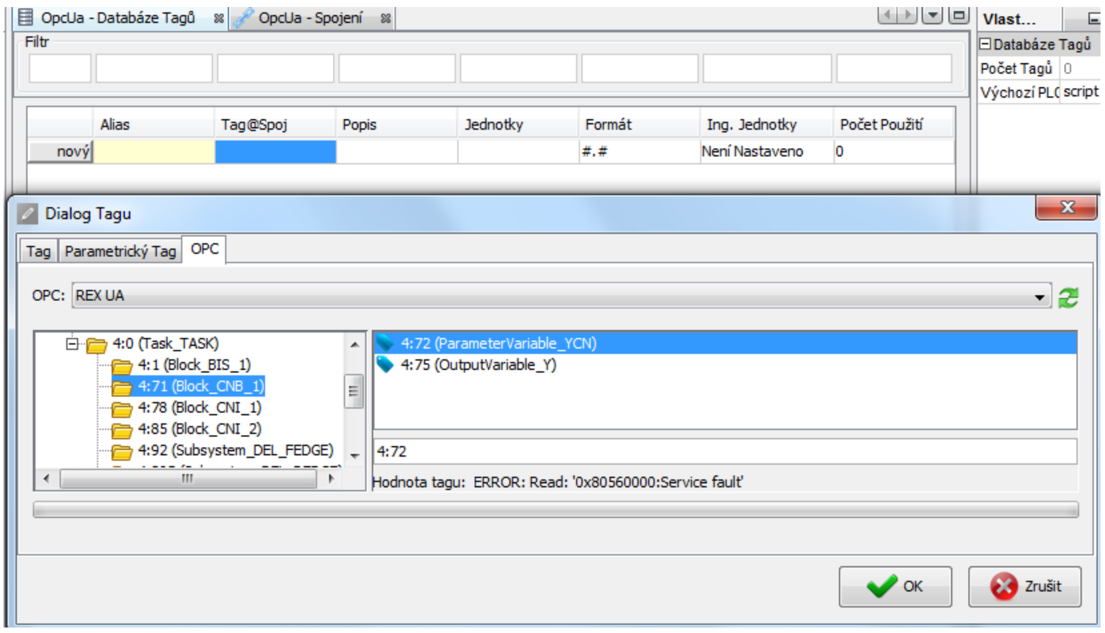
Cesta: opc.tcp://192.168.1.100:4888/None/None

Security police: None

MessSecMode: None

Figure 7.17: 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 7.18. The tag may be then used to show a value in HMI application, see picture 7.19. Download a final project to the device and use myView to show it.



	Alias	Tag@Spoj	Popis	Jednotky	Formát	Ing. Jednotky	Počet Použití
nový					#.#	Není Nastaveno	0

Databáze Tagů	
Počet Tagů	0
Výchozí PLC	script

OPC: REX UA

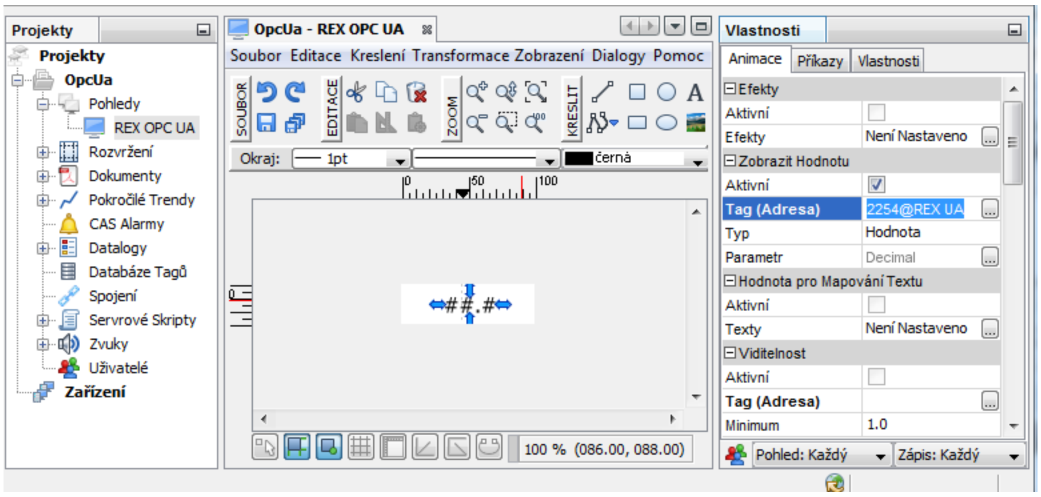
- 4:0 (Task_TASK)
  - 4:1 (Block_BIS_1)
  - 4:71 (Block_CNB_1)
  - 4:78 (Block_CNI_1)
  - 4:85 (Block_CNI_2)
  - 4:92 (Subsystem_DEL_FEDGE)

- 4:72 (ParameterVariable_YCN)
- 4:75 (OutputVariable_Y)

4:72

Hodnota tagu: ERROR: Read: '0x80560000:Service fault'

Figure 7.18: mySCADA: Tag creation



Efekty	
Aktivní	☐
Efekty	Není Nastaveno
Zobrazit Hodnotu	
Aktivní	☑
Tag (Adresa)	2254@REX UA
Typ	Hodnota
Parametr	Decimal
Hodnota pro Mapování Textu	
Aktivní	☐
Texty	Není Nastaveno
Viditelnost	
Aktivní	☐
Tag (Adresa)	
Minimum	1.0

Figure 7.19: 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 7.20. An error is indicated when a connection error occurs, authentication fails or a tag is not valid, see picture 7.25.

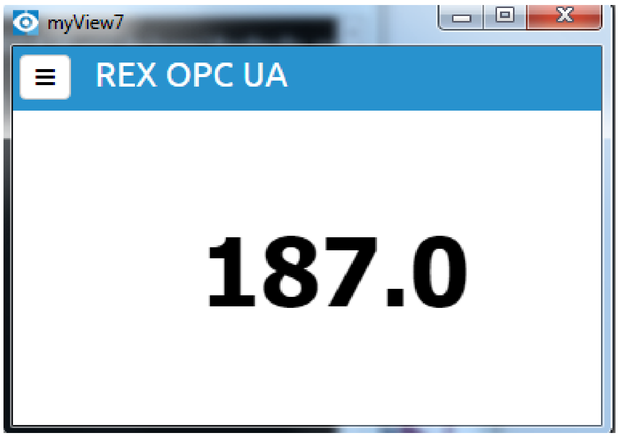


Figure 7.20: mySCADA: Running HMI

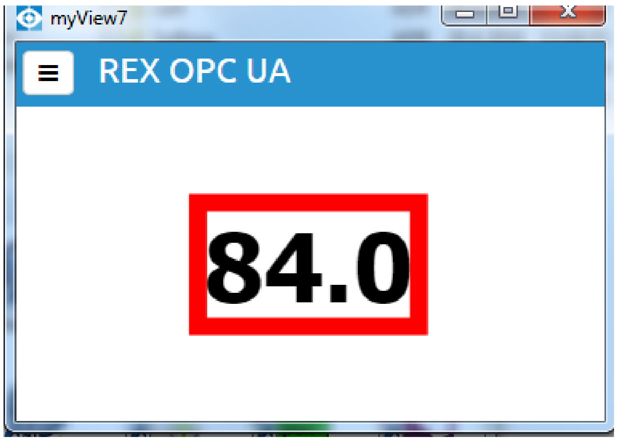
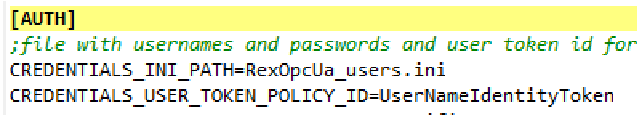


Figure 7.21: mySCADA: Data tag is not available

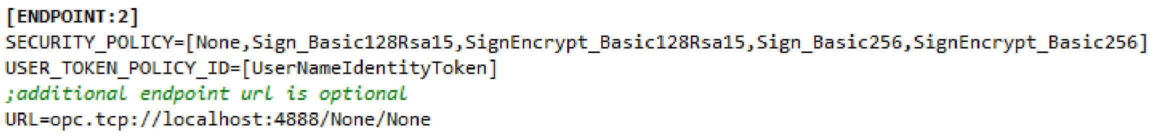
An endpoint has to have a `/None/None` suffix to mySCADA work properly with unencrypted connection. A policy ID for anonymous login must be set to zero, see picture 7.1.

To configure user name and password authentication policy a *UserNameIdentityToken* policy has to be set (see pictures 7.22 and 7.23) and user name and password must be supplied (see picture 7.24).



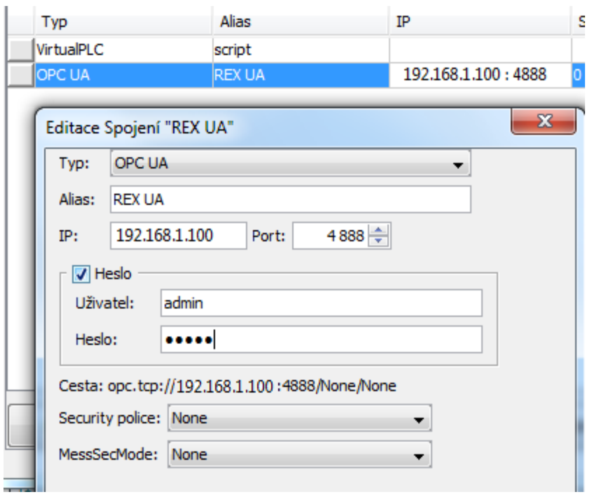
```
[AUTH]
;file with usernames and passwords and user token id for
CREDENTIALS_INI_PATH=RexOpcUa_users.ini
CREDENTIALS_USER_TOKEN_POLICY_ID=UserNameIdentityToken
```

Figure 7.22: 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 7.23: mySCADA: Endpoint policy settings



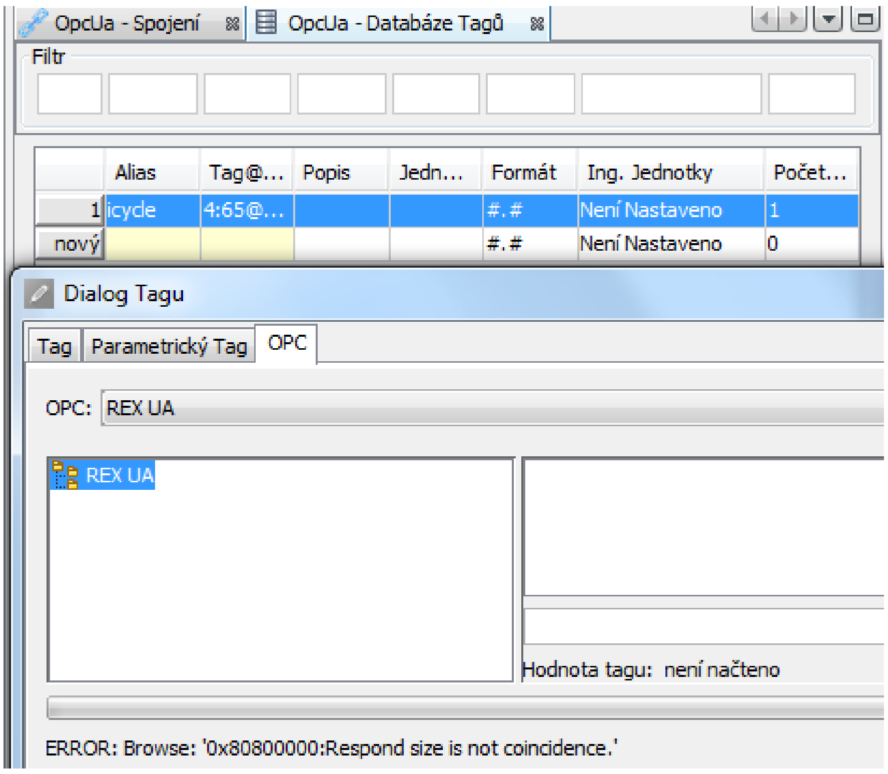
Typ	Alias	IP
VirtualPLC	script	
OPC UA	REX UA	192.168.1.100 : 4888

Editace Spojení "REX UA"

Typ: OPC UA
Alias: REX UA
IP: 192.168.1.100 Port: 4 888
[✓] Heslo
Uživatel: admin
Heslo: •••••
Cesta: opc.tcp://192.168.1.100 :4888/None/None
Security police: None
MessSecMode: None

Figure 7.24: mySCADA: Authentication using user name and password

An error is indicated when authentication fails, see picture 7.25.



	Alias	Tag@...	Popis	Jedn...	Formát	Ing. Jednotky	Počet...
1	icycle	4:65@...			#.#	Není Nastaveno	1
nový					#.#	Není Nastaveno	0

OPC: REX UA

REX UA

Hodnota tagu: není načteno

ERROR: Browse: '0x80800000:Respond size is not coincidence.'

Figure 7.25: mySCADA: Connection error

Bibliography

Documentation reference number: 17159