open62541

open62541 Documentation

Release 0.3.0-rc2

The open62541 authors

Nov 09, 2018

Contents

Introduction

open62541 (http://open62541.org) is an open source and free implementation of OPC UA (OPC Unified Architecture) written in the common subset of the C99 and C++98 languages. The library is usable with all major compilers and provides the necessary tools to implement dedicated OPC UA clients and servers, or to integrate OPC UA-based communication into existing applications. open62541 library is platform independent. All platform-specific functionality is implemented via exchangeable plugins. Plugin implementations are provided for the major operating systems.

open62541 is licensed under the Mozilla Public License v2.0. So the *open62541 library can be used in projects that are not open source*. Only changes to the open62541 library itself need to published under the same license. The plugins, as well as the server and client examples are in the public domain (CC0 license). They can be reused under any license and changes do not have to be published.

1.1 OPC Unified Architecture

OPC UA is a protocol for industrial communication and has been standardized in the IEC 62541 series. At its core, OPC UA defines

- an asynchronous *protocol* (built upon TCP, HTTP or SOAP) that defines the exchange of messages via sessions, (on top of) secure communication channels, (on top of) raw connections,

- a *type system* for protocol messages with a binary and XML-based encoding scheme,

- a meta-model for *information modeling*, that combines object-orientation with semantic triple-relations, and

- a set of 37 standard *services* to interact with server-side information models. The signature of each service is defined as a request and response message in the protocol type system.

The standard itself can be purchased from IEC or downloaded for free on the website of the OPC Foundation at https://opcfoundation.org/ (you need to register with a valid email).

The OPC Foundation drives the continuous improvement of the standard and the development of companion specifications. Companion specifications translate established concepts and reusable components from an application domain into OPC UA. They are created jointly with an established industry council or standardization body from the application domain. Furthermore, the OPC Foundation organizes events for the dissemination of the standard and provides the infrastructure and tools for compliance certification.

1.2 open62541 Features

open62541 implements the OPC UA binary protocol stack as well as a client and server SDK. It currently supports the Micro Embedded Device Server Profile plus some additional features. Server binaries can be well under 100kb in size, depending on the contained information model.

- Communication Stack

    - OPC UA binary protocol

    - Chunking (splitting of large messages)

    - Exchangeable network layer (plugin) for using custom networking APIs (e.g. on embedded targets)

    - Encrypted communication

    - Asynchronous service requests in the client

- Information model

    - Support for all OPC UA node types (including method nodes)

    - Support for adding and removing nodes and references also at runtime.

    - Support for inheritance and instantiation of object- and variable-types (custom constructor/destructor, instantiation of child nodes)

    - Access control for individual nodes

- Subscriptions

    - Support for subscriptions/monitoreditems for data change notifications

    - Very low resource consumption for each monitored value (event-based server architecture)

- Code-Generation

    - Support for generating data types from standard XML definitions

    - Support for generating server-side information models (nodesets) from standard XML definitions

Features on the roadmap for the 0.3 release series but missing in the initial v0.3 release are:

- Encrypted communication in the client

- Events (notifications emitted by objects, data change notifications are implemented)

- Event-loop (background tasks) in the client

1.3 Getting Help

For discussion and help besides this documentation, you can reach the open62541 community via

- the mailing list

- our IRC channel

- the bugtracker

1.4 Contributing

As an open source project, we invite new contributors to help improve open62541. Issue reports, bugfixes and new features are very welcome. The following are good starting points for new contributors:

- Report bugs

- Improve the documentation

• Work on issues marked as good first issue

CHAPTER 2

Building open62541

2.1 Building the Examples

Using the GCC compiler, the following calls build the examples on Linux.

```
cp /path-to/open62541.* . # copy single-file distribution to the local directory
cp /path-to/examples/server_variable.c . # copy the example server
gcc -std=c99 open62541.c server_variable.c -o server
```

2.2 Building the Library

2.2.1 Building with CMake on Ubuntu or Debian

```
sudo apt-get install git build-essential gcc pkg-config cmake python python-six

# enable additional features
sudo apt-get install cmake-curses-gui # for the ccmake graphical interface
sudo apt-get install libmbedtls-dev # for encryption support
sudo apt-get install check # for unit tests
sudo apt-get install python-sphinx graphviz # for documentation generation
sudo apt-get install python-sphinx-rtd-theme # documentation style

cd open62541
mkdir build
cd build
cmake ..
make

# select additional features
ccmake ..
make

# build documentation
make doc # html documentation
make doc_pdf # pdf documentation (requires LaTeX)
```

2.2.2 Building with CMake on Windows

Here we explain the build process for Visual Studio (2013 or newer). To build with MinGW, just replace the compiler selection in the call to CMake.

- Download and install
  - Python 2.7.x (Python 3.x works as well): https://python.org/downloads
  - Install python-six with the pip package manager (`pip install six`)
  - CMake: http://www.cmake.org/cmake/resources/software.html
  - Microsoft Visual Studio: https://www.visualstudio.com/products/visual-studio-community-vs
- Download the open62541 sources (using git or as a zipfile from github)
- Open a command shell (cmd) and run

```
cd <path-to>\open62541
mkdir build
cd build
<path-to>\cmake.exe .. -G "Visual Studio 14 2015"
:: You can use use cmake-gui for a graphical user-interface to select features
```

- Then open `buildopen62541.sln` in Visual Studio 2015 and build as usual

2.2.3 Building on OS X

- Download and install
  - Xcode: https://itunes.apple.com/us/app/xcode/id497799835?ls=1&mt=12
  - Homebrew: http://brew.sh/
  - Pip (a package manager for python, may be preinstalled): `sudo easy_install pip`
- Run the following in a shell

```
brew install cmake
pip install six # python 2/3 compatibility workarounds
pip install sphinx # for documentation generation
pip install sphinx_rtd_theme # documentation style
brew install graphviz # for graphics in the documentation
brew install check # for unit tests
brew install userspace-rcu # for multi-threading support
```

Follow Ubuntu instructions without the `apt-get` commands as these are taken care of by the above packages.

2.2.4 Building on OpenBSD

The procedure below works on OpenBSD 5.8 with gcc version 4.8.4, cmake version 3.2.3 and Python version 2.7.10.

- Install a recent gcc, python and cmake:

```
pkg_add gcc python cmake
```

- Tell the system to actually use the recent gcc (it gets installed as egcc on OpenBSD):

```
export CC=egcc CXX=eg++
```

- Now procede as described for Ubuntu/Debian:

```
cd open62541
mkdir build
cd build
cmake ..
make
```

2.3 Build Options

The open62541 project uses CMake to manage the build options, for code generation and to generate build projects for the different systems and IDEs. The tools *ccmake* or *cmake-gui* can be used to graphically set the build options.

Most options can be changed manually in `ua_config.h` (`open62541.h` for the single-file release) after the code generation. But usually there is no need to adjust them.

2.3.1 Build Type and Logging

CMAKE_BUILD_TYPE

- `RelWithDebInfo` -O2 optimization with debug symbols

- `Release` -O2 optimization without debug symbols

- `Debug` -O0 optimization with debug symbols

- `MinSizeRel` -Os optimization without debug symbols

UA_LOGLEVEL The SDK logs events of the level defined in `UA_LOGLEVEL` and above only. The logging event levels are as follows:

- 600: Fatal

- 500: Error

- 400: Warning

- 300: Info

- 200: Debug

- 100: Trace

2.3.2 UA_BUILD_* group

By default only the shared object libopen62541.so or the library open62541.dll and open62541.dll.a resp. open62541.lib are build. Additional artifacts can be specified by the following options:

UA_BUILD_EXAMPLES Compile example servers and clients from `examples/`*xyz*`.c`. A static and a dynamic binary is linked, respectively.

UA_BUILD_UNIT_TESTS Compile unit tests with Check framework. The tests can be executed with `make test`

UA_BUILD_EXAMPLES_NODESET_COMPILER Generate an OPC UA information model from a nodeset XML (experimental)

UA_BUILD_SELFSIGNED_CERTIFICATE Generate a self-signed certificate for the server (openSSL required)

2.3.3 UA_ENABLE_* group

This group contains build options related to the supported OPC UA features.

UA_ENABLE_SUBSCRIPTIONS Enable subscriptions

UA_ENABLE_METHODCALLS Enable the Method service set

UA_ENABLE_NODEMANAGEMENT Enable dynamic addition and removal of nodes at runtime

UA_ENABLE_AMALGAMATION Compile a single-file release into the files `open62541.c` and `open62541.h`

UA_ENABLE_MULTITHREADING Enable multi-threading support

UA_ENABLE_COVERAGE Measure the coverage of unit tests

UA_ENABLE_DISCOVERY Enable Discovery Service (LDS)

UA_ENABLE_DISCOVERY_MULTICAST Enable Discovery Service with multicast support (LDS-ME)

UA_ENABLE_DISCOVERY_SEMAPHORE Enable Discovery Semaphore support

Some options are marked as advanced. The advanced options need to be toggled to be visible in the cmake GUIs.

UA_ENABLE_TYPENAMES Add the type and member names to the UA_DataType structure. Enabled by default.

UA_ENABLE_STATUSCODE_DESCRIPTIONS Compile the human-readable name of the StatusCodes into the binary. Enabled by default.

UA_ENABLE_FULL_NS0 Use the full NS0 instead of a minimal Namespace 0 nodeset `UA_FILE_NS0` is used to specify the file for NS0 generation from namespace0 folder. Default value is `Opc.Ua.NodeSet2.xml`

UA_ENABLE_NONSTANDARD_UDP Enable udp extension

2.3.4 UA_DEBUG_* group

This group contains build options mainly useful for development of the library itself.

UA_DEBUG Enable assertions and additional definitions not intended for production builds

UA_DEBUG_DUMP_PKGS Dump every package received by the server as hexdump format

2.3.5 Building a shared library

open62541 is small enough that most users will want to statically link the library into their programs. If a shared library (.dll, .so) is required, this can be enabled in CMake with the `BUILD_SHARED_LIBS` option. Note that this option modifies the `ua_config.h` file that is also included in `open62541.h` for the single-file distribution.

2.3.6 Minimizing the binary size

The size of the generated binary can be reduced considerably by adjusting the build configuration. First, in CMake, the build type can be set to `CMAKE_BUILD_TYPE=MinSizeRel`. This sets the compiler flags to minimize the binary size. The build type also strips out debug information. Second, the binary size can be reduced by removing features via the build-flags described above.

Especially, logging takes up a lot of space in the binary and might not be needed in embedded scenarios. Setting `UA_LOGLEVEL` to a value above 600 (=FATAL) disables all logging. In addition, the feature-flags `UA_ENABLE_TYPENAMES` and `UA_ENABLE_STATUSCODE_DESCRIPTIONS` add static information to the binary that is only used for human-readable logging and debugging.

Tutorials

3.1 Working with Data Types

OPC UA defines a type system for values that can be encoded in the protocol messages. This tutorial shows some examples for available data types and their use. See the section on *Data Types* for the full definitions.

3.1.1 Basic Data Handling

This section shows the basic interaction patterns for data types. Make sure to compare with the type definitions in `ua_types.h`.

```c
#include <assert.h>
#include "open62541.h"

static void
variables_basic(void) {
    /* Int32 */
    UA_Int32 i = 5;
    UA_Int32 j;
    UA_Int32_copy(&i, &j);

    UA_Int32 *ip = UA_Int32_new();
    UA_Int32_copy(&i, ip);
    UA_Int32_delete(ip);

    /* String */
    UA_String s;
    UA_String_init(&s); /* _init zeroes out the entire memory of the datatype */
    char *test = "test";
    s.length = strlen(test);
    s.data = (UA_Byte*)test;

    UA_String s2;
    UA_String_copy(&s, &s2);
    UA_String_deleteMembers(&s2); /* Copying heap-allocated the dynamic content */

    UA_String s3 = UA_STRING("test2");
```

(continues on next page)

```
    UA_String s4 = UA_STRING_ALLOC("test2"); /* Copies the content to the heap */
    UA_Boolean eq = UA_String_equal(&s3, &s4);
    UA_String_deleteMembers(&s4);
    if(!eq)
        return;

    /* Structured Type */
    UA_CallRequest cr;
    UA_init(&cr, &UA_TYPES[UA_TYPES_CALLREQUEST]); /* Generic method */
    UA_CallRequest_init(&cr); /* Shorthand for the previous line */

    cr.requestHeader.timestamp = UA_DateTime_now(); /* Members of a structure */

    cr.methodsToCall = (UA_CallMethodRequest *)UA_Array_new(5, &UA_TYPES[UA_TYPES_
→CALLMETHODREQUEST]);
    cr.methodsToCallSize = 5; /* Array size needs to be made known */

    UA_CallRequest *cr2 = UA_CallRequest_new();
    UA_copy(&cr, cr2, &UA_TYPES[UA_TYPES_CALLREQUEST]);
    UA_CallRequest_deleteMembers(&cr);
    UA_CallRequest_delete(cr2);
}
```

3.1.2 NodeIds

An OPC UA information model is made up of nodes and references between nodes. Every node has a unique *NodeId*. NodeIds refer to a namespace with an additional identifier value that can be an integer, a string, a guid or a bytestring.

```
static void
variables_nodeids(void) {
    UA_NodeId id1 = UA_NODEID_NUMERIC(1, 1234);
    id1.namespaceIndex = 3;

    UA_NodeId id2 = UA_NODEID_STRING(1, "testid"); /* the string is static */
    UA_Boolean eq = UA_NodeId_equal(&id1, &id2);
    if(eq)
        return;

    UA_NodeId id3;
    UA_NodeId_copy(&id2, &id3);
    UA_NodeId_deleteMembers(&id3);

    UA_NodeId id4 = UA_NODEID_STRING_ALLOC(1, "testid"); /* the string is copied
                                                            to the heap */
    UA_NodeId_deleteMembers(&id4);
}
```

3.1.3 Variants

The datatype *Variant* belongs to the built-in datatypes of OPC UA and is used as a container type. A variant can hold any other datatype as a scalar (except variant) or as an array. Array variants can additionally denote the dimensionality of the data (e.g. a 2x3 matrix) in an additional integer array.

```
static void
variables_variants(void) {
    /* Set a scalar value */
```

```
    UA_Variant v;
    UA_Int32 i = 42;
    UA_Variant_setScalar(&v, &i, &UA_TYPES[UA_TYPES_INT32]);

    /* Make a copy */
    UA_Variant v2;
    UA_Variant_copy(&v, &v2);
    UA_Variant_deleteMembers(&v2);

    /* Set an array value */
    UA_Variant v3;
    UA_Double d[9] = {1.0, 2.0, 3.0,
                      4.0, 5.0, 6.0,
                      7.0, 8.0, 9.0};
    UA_Variant_setArrayCopy(&v3, d, 9, &UA_TYPES[UA_TYPES_DOUBLE]);

    /* Set array dimensions */
    v3.arrayDimensions = (UA_UInt32 *)UA_Array_new(2, &UA_TYPES[UA_TYPES_UINT32]);
    v3.arrayDimensionsSize = 2;
    v3.arrayDimensions[0] = 3;
    v3.arrayDimensions[1] = 3;
    UA_Variant_deleteMembers(&v3);
}
```

It follows the main function, making use of the above definitions.

```
int main(void) {
    variables_basic();
    variables_nodeids();
    variables_variants();
    return 0;
}
```

3.2 Building a Simple Server

This series of tutorial guide you through your first steps with open62541. For compiling the examples, you need a compiler (MS Visual Studio 2015 or newer, GCC, Clang and MinGW32 are all known to be working). The compilation instructions are given for GCC but should be straightforward to adapt.

It will also be very helpful to install an OPC UA Client with a graphical frontend, such as UAExpert by Unified Automation. That will enable you to examine the information model of any OPC UA server.

To get started, downdload the open62541 single-file release from http://open62541.org or generate it according to the *build instructions* with the "amalgamation" option enabled. From now on, we assume you have the open62541.c/.h files in the current folder. Now create a new C source-file called myServer.c with the following content:

```
#include <signal.h>
#include "open62541.h"

UA_Boolean running = true;
static void stopHandler(int sig) {
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_USERLAND, "received ctrl-c");
    running = false;
}

int main(void) {
    signal(SIGINT, stopHandler);
    signal(SIGTERM, stopHandler);
```

```
    UA_ServerConfig *config = UA_ServerConfig_new_default();
    UA_Server *server = UA_Server_new(config);

    UA_StatusCode retval = UA_Server_run(server, &running);
    UA_Server_delete(server);
    UA_ServerConfig_delete(config);
    return (int)retval;
}
```

This is all that is needed for a simple OPC UA server. With the GCC compiler, the following command produces an executable:

```
$ gcc -std=c99 open62541.c myServer.c -o myServer
```

In a MinGW environment, the Winsock library must be added.

```
$ gcc -std=c99 open62541.c myServer.c -lws2_32 -o myServer.exe
```

Now start the server (stop with ctrl-c):

```
$ ./myServer
```

You have now compiled and run your first OPC UA server. You can go ahead and browse the information model with client. The server is listening on `opc.tcp://localhost:4840`. In the next two sections, we will continue to explain the different parts of the code in detail.

3.2.1 Server Configuration and Plugins

open62541 provides a flexible framework for building OPC UA servers and clients. The goals is to have a core library that accommodates for all use cases and runs on all platforms. Users can then adjust the library to fit their use case via configuration and by developing (platform-specific) plugins. The core library is based on C99 only and does not even require basic POSIX support. For example, the lowlevel networking code is implemented as an exchangeable plugin. But don't worry. *open62541* provides plugin implementations for most platforms and sensible default configurations out-of-the-box.

In the above server code, we simply take the default server configuration and add a single TCP network layer that is listening on port 4840.

3.2.2 Server Lifecycle

The code in this example shows the three parts for server lifecycle management: Creating a server, running the server, and deleting the server. Creating and deleting a server is trivial once the configuration is set up. The server is started with `UA_Server_run`. Internally, the server then uses timeouts to schedule regular tasks. Between the timeouts, the server listens on the network layer for incoming messages.

You might ask how the server knows when to stop running. For this, we have created a global variable `running`. Furthermore, we have registered the method `stopHandler` that catches the signal (interrupt) the program receives when the operating systems tries to close it. This happens for example when you press ctrl-c in a terminal program. The signal handler then sets the variable `running` to false and the server shuts down once it takes back control.[1]

In order to integrated OPC UA in a single-threaded application with its own mainloop (for example provided by a GUI toolkit), one can alternatively drive the server manually. See the section of the server documentation on *Server Lifecycle* for details.

The server configuration and lifecycle management is needed for all servers. We will use it in the following tutorials without further comment.

---

[1] Be careful with global variables in multi-threaded applications. You might want to allocate the `running` variable on the heap.

3.3 Adding Variables to a Server

This tutorial shows how to work with data types and how to add variable nodes to a server. First, we add a new variable to the server. Take a look at the definition of the `UA_VariableAttrbitues` structure to see the list of all attributes defined for VariableNodes.

Note that the default settings have the AccessLevel of the variable value as read only. See below for making the variable writable.

```c
#include <signal.h>
#include <stdio.h>
#include "open62541.h"

static void
addVariable(UA_Server *server) {
    /* Define the attribute of the myInteger variable node */
    UA_VariableAttributes attr = UA_VariableAttributes_default;
    UA_Int32 myInteger = 42;
    UA_Variant_setScalar(&attr.value, &myInteger, &UA_TYPES[UA_TYPES_INT32]);
    attr.description = UA_LOCALIZEDTEXT("en-US","the answer");
    attr.displayName = UA_LOCALIZEDTEXT("en-US","the answer");
    attr.dataType = UA_TYPES[UA_TYPES_INT32].typeId;
    attr.accessLevel = UA_ACCESSLEVELMASK_READ | UA_ACCESSLEVELMASK_WRITE;

    /* Add the variable node to the information model */
    UA_NodeId myIntegerNodeId = UA_NODEID_STRING(1, "the.answer");
    UA_QualifiedName myIntegerName = UA_QUALIFIEDNAME(1, "the answer");
    UA_NodeId parentNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER);
    UA_NodeId parentReferenceNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_ORGANIZES);
    UA_Server_addVariableNode(server, myIntegerNodeId, parentNodeId,
                              parentReferenceNodeId, myIntegerName,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_BASEDATAVARIABLETYPE),
→attr, NULL, NULL);
}
```

Now we change the value with the write service. This uses the same service implementation that can also be reached over the network by an OPC UA client.

```c
static void
writeVariable(UA_Server *server) {
    UA_NodeId myIntegerNodeId = UA_NODEID_STRING(1, "the.answer");

    /* Write a different integer value */
    UA_Int32 myInteger = 43;
    UA_Variant myVar;
    UA_Variant_init(&myVar);
    UA_Variant_setScalar(&myVar, &myInteger, &UA_TYPES[UA_TYPES_INT32]);
    UA_Server_writeValue(server, myIntegerNodeId, myVar);

    /* Set the status code of the value to an error code. The function
     * UA_Server_write provides access to the raw service. The above
     * UA_Server_writeValue is syntactic sugar for writing a specific node
     * attribute with the write service. */
    UA_WriteValue wv;
    UA_WriteValue_init(&wv);
    wv.nodeId = myIntegerNodeId;
    wv.attributeId = UA_ATTRIBUTEID_VALUE;
    wv.value.status = UA_STATUSCODE_BADNOTCONNECTED;
    wv.value.hasStatus = true;
    UA_Server_write(server, &wv);

    /* Reset the variable to a good statuscode with a value */
```

```
    wv.value.hasStatus = false;
    wv.value.value = myVar;
    wv.value.hasValue = true;
    UA_Server_write(server, &wv);
}
```

Note how we initially set the DataType attribute of the variable node to the NodeId of the Int32 data type. This forbids writing values that are not an Int32. The following code shows how this consistency check is performed for every write.

```
static void
writeWrongVariable(UA_Server *server) {
    UA_NodeId myIntegerNodeId = UA_NODEID_STRING(1, "the.answer");

    /* Write a string */
    UA_String myString = UA_STRING("test");
    UA_Variant myVar;
    UA_Variant_init(&myVar);
    UA_Variant_setScalar(&myVar, &myString, &UA_TYPES[UA_TYPES_STRING]);
    UA_StatusCode retval = UA_Server_writeValue(server, myIntegerNodeId, myVar);
    printf("Writing a string returned statuscode %s\n", UA_StatusCode_
→name(retval));
}
```

It follows the main server code, making use of the above definitions.

```
UA_Boolean running = true;
static void stopHandler(int sign) {
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_SERVER, "received ctrl-c");
    running = false;
}

int main(void) {
    signal(SIGINT, stopHandler);
    signal(SIGTERM, stopHandler);

    UA_ServerConfig *config = UA_ServerConfig_new_default();
    UA_Server *server = UA_Server_new(config);

    addVariable(server);
    writeVariable(server);
    writeWrongVariable(server);

    UA_StatusCode retval = UA_Server_run(server, &running);
    UA_Server_delete(server);
    UA_ServerConfig_delete(config);
    return (int)retval;
}
```

3.4 Connecting a Variable with a Physical Process

In OPC UA-based architectures, servers are typically situated near the source of information. In an industrial context, this translates into servers being near the physical process and clients consuming the data at runtime. In the previous tutorial, we saw how to add variables to an OPC UA information model. This tutorial shows how to connect a variable to runtime information, for example from measurements of a physical process. For simplicity, we take the system clock as the underlying "process".

The following code snippets are each concerned with a different way of updating variable values at runtime. Taken together, the code snippets define a compilable source file.

3.4.1 Updating variables manually

As a starting point, assume that a variable for a value of type *DateTime* has been created in the server with the identifier "ns=1,s=current-time". Assuming that our applications gets triggered when a new value arrives from the underlying process, we can just write into the variable.

```c
#include <signal.h>
#include "open62541.h"

static void
updateCurrentTime(UA_Server *server) {
    UA_DateTime now = UA_DateTime_now();
    UA_Variant value;
    UA_Variant_setScalar(&value, &now, &UA_TYPES[UA_TYPES_DATETIME]);
    UA_NodeId currentNodeId = UA_NODEID_STRING(1, "current-time");
    UA_Server_writeValue(server, currentNodeId, value);
}

static void
addCurrentTimeVariable(UA_Server *server) {
    UA_DateTime now = 0;
    UA_VariableAttributes attr = UA_VariableAttributes_default;
    attr.displayName = UA_LOCALIZEDTEXT("en-US", "Current time");
    attr.accessLevel = UA_ACCESSLEVELMASK_READ | UA_ACCESSLEVELMASK_WRITE;
    UA_Variant_setScalar(&attr.value, &now, &UA_TYPES[UA_TYPES_DATETIME]);

    UA_NodeId currentNodeId = UA_NODEID_STRING(1, "current-time");
    UA_QualifiedName currentName = UA_QUALIFIEDNAME(1, "current-time");
    UA_NodeId parentNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER);
    UA_NodeId parentReferenceNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_ORGANIZES);
    UA_NodeId variableTypeNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_
↪BASEDATAVARIABLETYPE);
    UA_Server_addVariableNode(server, currentNodeId, parentNodeId,
                              parentReferenceNodeId, currentName,
                              variableTypeNodeId, attr, NULL, NULL);

    updateCurrentTime(server);
}
```

3.4.2 Variable Value Callback

When a value changes continuously, such as the system time, updating the value in a tight loop would take up a lot of resources. Value callbacks allow to synchronize a variable value with an external representation. They attach callbacks to the variable that are executed before every read and after every write operation.

```c
static void
beforeReadTime(UA_Server *server,
               const UA_NodeId *sessionId, void *sessionContext,
               const UA_NodeId *nodeid, void *nodeContext,
               const UA_NumericRange *range, const UA_DataValue *data) {
    UA_DateTime now = UA_DateTime_now();
    UA_Variant value;
    UA_Variant_setScalar(&value, &now, &UA_TYPES[UA_TYPES_DATETIME]);
    UA_NodeId currentNodeId = UA_NODEID_STRING(1, "current-time");
    UA_Server_writeValue(server, currentNodeId, value);
}

static void
afterWriteTime(UA_Server *server,
               const UA_NodeId *sessionId, void *sessionContext,
```

(continues on next page)

```c
                     const UA_NodeId *nodeId, void *nodeContext,
                     const UA_NumericRange *range, const UA_DataValue *data) {
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_USERLAND,
                "The variable was updated");
}

static void
addValueCallbackToCurrentTimeVariable(UA_Server *server) {
    UA_NodeId currentNodeId = UA_NODEID_STRING(1, "current-time");
    UA_ValueCallback callback ;
    callback.onRead = beforeReadTime;
    callback.onWrite = afterWriteTime;
    UA_Server_setVariableNode_valueCallback(server, currentNodeId, callback);
}
```

3.4.3 Variable Data Sources

With value callbacks, the value is still stored in the variable node. So-called data sources go one step further. The server redirects every read and write request to a callback function. Upon reading, the callback provides copy of the current value. Internally, the data source needs to implement its own memory management.

```c
static UA_StatusCode
readCurrentTime(UA_Server *server,
                const UA_NodeId *sessionId, void *sessionContext,
                const UA_NodeId *nodeId, void *nodeContext,
                UA_Boolean sourceTimeStamp, const UA_NumericRange *range,
                UA_DataValue *dataValue) {
    UA_DateTime now = UA_DateTime_now();
    UA_Variant_setScalarCopy(&dataValue->value, &now,
                             &UA_TYPES[UA_TYPES_DATETIME]);
    dataValue->hasValue = true;
    return UA_STATUSCODE_GOOD;
}

static UA_StatusCode
writeCurrentTime(UA_Server *server,
                 const UA_NodeId *sessionId, void *sessionContext,
                 const UA_NodeId *nodeId, void *nodeContext,
                 const UA_NumericRange *range, const UA_DataValue *data) {
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_USERLAND,
                "Changing the system time is not implemented");
    return UA_STATUSCODE_BADINTERNALERROR;
}

static void
addCurrentTimeDataSourceVariable(UA_Server *server) {
    UA_VariableAttributes attr = UA_VariableAttributes_default;
    attr.displayName = UA_LOCALIZEDTEXT("en-US", "Current time - data source");
    attr.accessLevel = UA_ACCESSLEVELMASK_READ | UA_ACCESSLEVELMASK_WRITE;

    UA_NodeId currentNodeId = UA_NODEID_STRING(1, "current-time-datasource");
    UA_QualifiedName currentName = UA_QUALIFIEDNAME(1, "current-time-datasource");
    UA_NodeId parentNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER);
    UA_NodeId parentReferenceNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_ORGANIZES);
    UA_NodeId variableTypeNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_
→BASEDATAVARIABLETYPE);

    UA_DataSource timeDataSource;
    timeDataSource.read = readCurrentTime;
```

```
    timeDataSource.write = writeCurrentTime;
    UA_Server_addDataSourceVariableNode(server, currentNodeId, parentNodeId,
                                        parentReferenceNodeId, currentName,
                                        variableTypeNodeId, attr,
                                        timeDataSource, NULL, NULL);
}
```

It follows the main server code, making use of the above definitions.

```
UA_Boolean running = true;
static void stopHandler(int sign) {
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_SERVER, "received ctrl-c");
    running = false;
}

int main(void) {
    signal(SIGINT, stopHandler);
    signal(SIGTERM, stopHandler);

    UA_ServerConfig *config = UA_ServerConfig_new_default();
    UA_Server *server = UA_Server_new(config);

    addCurrentTimeVariable(server);
    addValueCallbackToCurrentTimeVariable(server);
    addCurrentTimeDataSourceVariable(server);

    UA_StatusCode retval = UA_Server_run(server, &running);
    UA_Server_delete(server);
    UA_ServerConfig_delete(config);
    return (int)retval;
}
```

3.4.4 DataChange Notifications

A client that is interested in the current value of a variable does not need to regularly poll the variable. Instead, he can use the Subscription mechanism to be notified about changes.

Within a Subscription, the client adds so-called MonitoredItems. A DataChange MonitoredItem defines a node attribute (usually the value attribute) that is monitored for changes. The server internally reads the value in the defined interval and generates the appropriate notifications. The three ways of updating node values discussed above are all usable in combination with notifications. That is because notifications use the standard *Read* service to look for value changes.

3.5 Working with Variable Types

Variable types have three functions:

- Constrain the possible data type, value rank and array dimensions of the variables of that type. This allows interface code to be written against the generic type definition, so it is applicable for all instances.

- Provide a sensible default value

- Enable a semantic interpretation of the variable based on its type

In the example of this tutorial, we represent a point in 2D space by an array of double values. The following function adds the corresponding VariableTypeNode to the hierarchy of variable types.

```
#include <signal.h>
#include "open62541.h"


static UA_NodeId pointTypeId;

static void
addVariableType2DPoint(UA_Server *server) {
    UA_VariableTypeAttributes vtAttr = UA_VariableTypeAttributes_default;
    vtAttr.dataType = UA_TYPES[UA_TYPES_DOUBLE].typeId;
    vtAttr.valueRank = UA_VALUERANK_ONE_DIMENSION;
    UA_UInt32 arrayDims[1] = {2};
    vtAttr.arrayDimensions = arrayDims;
    vtAttr.arrayDimensionsSize = 1;
    vtAttr.displayName = UA_LOCALIZEDTEXT("en-US", "2DPoint Type");

    /* a matching default value is required */
    UA_Double zero[2] = {0.0, 0.0};
    UA_Variant_setArray(&vtAttr.value, zero, 2, &UA_TYPES[UA_TYPES_DOUBLE]);

    UA_Server_addVariableTypeNode(server, UA_NODEID_NULL,
                                  UA_NODEID_NUMERIC(0, UA_NS0ID_
→BASEDATAVARIABLETYPE),
                                  UA_NODEID_NUMERIC(0, UA_NS0ID_HASSUBTYPE),
                                  UA_QUALIFIEDNAME(1, "2DPoint Type"), UA_NODEID_
→NULL,
                                  vtAttr, NULL, &pointTypeId);
}
```

Now the new variable type for *2DPoint* can be referenced during the creation of a new variable. If no value is given, the default from the variable type is copied during instantiation.

```
static UA_NodeId pointVariableId;

static void
addVariable(UA_Server *server) {
    /* Prepare the node attributes */
    UA_VariableAttributes vAttr = UA_VariableAttributes_default;
    vAttr.dataType = UA_TYPES[UA_TYPES_DOUBLE].typeId;
    vAttr.valueRank = UA_VALUERANK_ONE_DIMENSION;
    UA_UInt32 arrayDims[1] = {2};
    vAttr.arrayDimensions = arrayDims;
    vAttr.arrayDimensionsSize = 1;
    vAttr.displayName = UA_LOCALIZEDTEXT("en-US", "2DPoint Variable");
    vAttr.accessLevel = UA_ACCESSLEVELMASK_READ | UA_ACCESSLEVELMASK_WRITE;
    /* vAttr.value is left empty, the server instantiates with the default value */

    /* Add the node */
    UA_Server_addVariableNode(server, UA_NODEID_NULL,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT),
                              UA_QUALIFIEDNAME(1, "2DPoint Type"), pointTypeId,
                              vAttr, NULL, &pointVariableId);
}
```

The constraints of the variable type are enforced when creating new variable instances of the type. In the following function, adding a variable of *2DPoint* type with a string value fails because The value does not match the variable type constraints.

```
static void
addVariableFail(UA_Server *server) {
    /* Prepare the node attributes */
    UA_VariableAttributes vAttr = UA_VariableAttributes_default;
```

(continues on next page)

```c
    vAttr.dataType = UA_TYPES[UA_TYPES_DOUBLE].typeId;
    vAttr.valueRank = UA_VALUERANK_SCALAR; /* a scalar. this is not allowed per␣
→the variable type */
    vAttr.displayName = UA_LOCALIZEDTEXT("en-US", "2DPoint Variable (fail)");
    UA_String s = UA_STRING("2dpoint?");
    UA_Variant_setScalar(&vAttr.value, &s, &UA_TYPES[UA_TYPES_STRING]);

    /* Add the node will return UA_STATUSCODE_BADTYPEMISMATCH*/
    UA_Server_addVariableNode(server, UA_NODEID_NULL,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT),
                              UA_QUALIFIEDNAME(1, "2DPoint Type (fail)"),␣
→pointTypeId,
                              vAttr, NULL, NULL);
}
```

The constraints of the variable type are enforced when writing the datatype, valuerank and arraydimensions attributes of the variable. This, in turn, constrains the value attribute of the variable.

```c
static void
writeVariable(UA_Server *server) {
    UA_StatusCode retval = UA_Server_writeValueRank(server, pointVariableId, UA_
→VALUERANK_ONE_OR_MORE_DIMENSIONS);
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_USERLAND,
                "Setting the Value Rank failed with Status Code %s",
                UA_StatusCode_name(retval));

}
```

It follows the main server code, making use of the above definitions.

```c
UA_Boolean running = true;
static void stopHandler(int sign) {
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_SERVER, "received ctrl-c");
    running = false;
}

int main(void) {
    signal(SIGINT, stopHandler);
    signal(SIGTERM, stopHandler);

    UA_ServerConfig *config = UA_ServerConfig_new_default();
    UA_Server *server = UA_Server_new(config);

    addVariableType2DPoint(server);
    addVariable(server);
    addVariableFail(server);
    writeVariable(server);

    UA_StatusCode retval = UA_Server_run(server, &running);
    UA_Server_delete(server);
    UA_ServerConfig_delete(config);
    return (int)retval;
}
```

3.6 Working with Objects and Object Types

3.6.1 Using objects to structure information models

Assume a situation where we want to model a set of pumps and their runtime state in an OPC UA information model. Of course, all pump representations should follow the same basic structure, For example, we might have graphical representation of pumps in a SCADA visualisation that shall be resuable for all pumps.

Following the object-oriented programming paradigm, every pump is represented by an object with the following layout:

The following code manually defines a pump and its member variables. We omit setting constraints on the variable values as this is not the focus of this tutorial and was already covered.

```c
#include <signal.h>
#include "open62541.h"

static void
manuallyDefinePump(UA_Server *server) {
    UA_NodeId pumpId; /* get the nodeid assigned by the server */
    UA_ObjectAttributes oAttr = UA_ObjectAttributes_default;
    oAttr.displayName = UA_LOCALIZEDTEXT("en-US", "Pump (Manual)");
    UA_Server_addObjectNode(server, UA_NODEID_NULL,
```

```
                                 UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER),
                                 UA_NODEID_NUMERIC(0, UA_NS0ID_ORGANIZES),
                                 UA_QUALIFIEDNAME(1, "Pump (Manual)"), UA_NODEID_
→NUMERIC(0, UA_NS0ID_BASEOBJECTTYPE),
                                 oAttr, NULL, &pumpId);

    UA_VariableAttributes mnAttr = UA_VariableAttributes_default;
    UA_String manufacturerName = UA_STRING("Pump King Ltd.");
    UA_Variant_setScalar(&mnAttr.value, &manufacturerName, &UA_TYPES[UA_TYPES_
→STRING]);
    mnAttr.displayName = UA_LOCALIZEDTEXT("en-US", "ManufacturerName");
    UA_Server_addVariableNode(server, UA_NODEID_NULL, pumpId,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT),
                              UA_QUALIFIEDNAME(1, "ManufacturerName"),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_BASEDATAVARIABLETYPE),
→mnAttr, NULL, NULL);

    UA_VariableAttributes modelAttr = UA_VariableAttributes_default;
    UA_String modelName = UA_STRING("Mega Pump 3000");
    UA_Variant_setScalar(&modelAttr.value, &modelName, &UA_TYPES[UA_TYPES_STRING]);
    modelAttr.displayName = UA_LOCALIZEDTEXT("en-US", "ModelName");
    UA_Server_addVariableNode(server, UA_NODEID_NULL, pumpId,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT),
                              UA_QUALIFIEDNAME(1, "ModelName"),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_BASEDATAVARIABLETYPE),
→modelAttr, NULL, NULL);

    UA_VariableAttributes statusAttr = UA_VariableAttributes_default;
    UA_Boolean status = true;
    UA_Variant_setScalar(&statusAttr.value, &status, &UA_TYPES[UA_TYPES_BOOLEAN]);
    statusAttr.displayName = UA_LOCALIZEDTEXT("en-US", "Status");
    UA_Server_addVariableNode(server, UA_NODEID_NULL, pumpId,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT),
                              UA_QUALIFIEDNAME(1, "Status"),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_BASEDATAVARIABLETYPE),
→statusAttr, NULL, NULL);

    UA_VariableAttributes rpmAttr = UA_VariableAttributes_default;
    UA_Double rpm = 50.0;
    UA_Variant_setScalar(&rpmAttr.value, &rpm, &UA_TYPES[UA_TYPES_DOUBLE]);
    rpmAttr.displayName = UA_LOCALIZEDTEXT("en-US", "MotorRPM");
    UA_Server_addVariableNode(server, UA_NODEID_NULL, pumpId,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT),
                              UA_QUALIFIEDNAME(1, "MotorRPMs"),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_BASEDATAVARIABLETYPE),
→rpmAttr, NULL, NULL);
}
```

3.6.2 Object types, type hierarchies and instantiation

Building up each object manually requires us to write a lot of code. Furthermore, there is no way for clients to detect that an object represents a pump. (We might use naming conventions or similar to detect pumps. But that's not exactly a clean solution.) Furthermore, we might have more devices than just pumps. And we require all devices to share some common structure. The solution is to define ObjectTypes in a hierarchy with inheritance relations.

Children that are marked mandatory are automatically instantiated together with the parent object. This is indicated by a *hasModellingRule* reference to an object that representes the *mandatory* modelling rule.

```c
/* predefined identifier for later use */
UA_NodeId pumpTypeId = {1, UA_NODEIDTYPE_NUMERIC, {1001}};

static void
defineObjectTypes(UA_Server *server) {
    /* Define the object type for "Device" */
    UA_NodeId deviceTypeId; /* get the nodeid assigned by the server */
    UA_ObjectTypeAttributes dtAttr = UA_ObjectTypeAttributes_default;
    dtAttr.displayName = UA_LOCALIZEDTEXT("en-US", "DeviceType");
    UA_Server_addObjectTypeNode(server, UA_NODEID_NULL,
                                UA_NODEID_NUMERIC(0, UA_NS0ID_BASEOBJECTTYPE),
                                UA_NODEID_NUMERIC(0, UA_NS0ID_HASSUBTYPE),
                                UA_QUALIFIEDNAME(1, "DeviceType"), dtAttr,
                                NULL, &deviceTypeId);

    UA_VariableAttributes mnAttr = UA_VariableAttributes_default;
    mnAttr.displayName = UA_LOCALIZEDTEXT("en-US", "ManufacturerName");
    UA_NodeId manufacturerNameId;
    UA_Server_addVariableNode(server, UA_NODEID_NULL, deviceTypeId,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT),
```

(continues on next page)

```
                              UA_QUALIFIEDNAME(1, "ManufacturerName"),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_BASEDATAVARIABLETYPE),␣
→mnAttr, NULL, &manufacturerNameId);
    /* Make the manufacturer name mandatory */
    UA_Server_addReference(server, manufacturerNameId,
                           UA_NODEID_NUMERIC(0, UA_NS0ID_HASMODELLINGRULE),
                           UA_EXPANDEDNODEID_NUMERIC(0, UA_NS0ID_MODELLINGRULE_
→MANDATORY), true);


    UA_VariableAttributes modelAttr = UA_VariableAttributes_default;
    modelAttr.displayName = UA_LOCALIZEDTEXT("en-US", "ModelName");
    UA_Server_addVariableNode(server, UA_NODEID_NULL, deviceTypeId,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT),
                              UA_QUALIFIEDNAME(1, "ModelName"),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_BASEDATAVARIABLETYPE),␣
→modelAttr, NULL, NULL);

    /* Define the object type for "Pump" */
    UA_ObjectTypeAttributes ptAttr = UA_ObjectTypeAttributes_default;
    ptAttr.displayName = UA_LOCALIZEDTEXT("en-US", "PumpType");
    UA_Server_addObjectTypeNode(server, pumpTypeId,
                                deviceTypeId, UA_NODEID_NUMERIC(0, UA_NS0ID_
→HASSUBTYPE),
                                UA_QUALIFIEDNAME(1, "PumpType"), ptAttr,
                                NULL, NULL);

    UA_VariableAttributes statusAttr = UA_VariableAttributes_default;
    statusAttr.displayName = UA_LOCALIZEDTEXT("en-US", "Status");
    statusAttr.valueRank = UA_VALUERANK_SCALAR;
    UA_NodeId statusId;
    UA_Server_addVariableNode(server, UA_NODEID_NULL, pumpTypeId,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT),
                              UA_QUALIFIEDNAME(1, "Status"),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_BASEDATAVARIABLETYPE),␣
→statusAttr, NULL, &statusId);
    /* Make the status variable mandatory */
    UA_Server_addReference(server, statusId,
                           UA_NODEID_NUMERIC(0, UA_NS0ID_HASMODELLINGRULE),
                           UA_EXPANDEDNODEID_NUMERIC(0, UA_NS0ID_MODELLINGRULE_
→MANDATORY), true);

    UA_VariableAttributes rpmAttr = UA_VariableAttributes_default;
    rpmAttr.displayName = UA_LOCALIZEDTEXT("en-US", "MotorRPM");
    rpmAttr.valueRank = UA_VALUERANK_SCALAR;
    UA_Server_addVariableNode(server, UA_NODEID_NULL, pumpTypeId,
                              UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT),
                              UA_QUALIFIEDNAME(1, "MotorRPMs"),
                              UA_NODEID_NUMERIC(0, UA_NS0ID_BASEDATAVARIABLETYPE),␣
→rpmAttr, NULL, NULL);
}
```

Now we add the derived ObjectType for the pump that inherits from the device object type. The resulting object contains all mandatory child variables. These are simply copied over from the object type. The object has a reference of type `hasTypeDefinition` to the object type, so that clients can detect the type-instance relation at runtime.

```
static void
addPumpObjectInstance(UA_Server *server, char *name) {
    UA_ObjectAttributes oAttr = UA_ObjectAttributes_default;
    oAttr.displayName = UA_LOCALIZEDTEXT("en-US", name);
```

```
    UA_Server_addObjectNode(server, UA_NODEID_NULL,
                            UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER),
                            UA_NODEID_NUMERIC(0, UA_NS0ID_ORGANIZES),
                            UA_QUALIFIEDNAME(1, name),
                            pumpTypeId, /* this refers to the object type
                                           identifier */
                            oAttr, NULL, NULL);
}
```

Often times, we want to run a constructor function on a new object. This is especially useful when an object is instantiated at runtime (with the AddNodes service) and the integration with an underlying process canot be manually defined. In the following constructor example, we simply set the pump status to on.

```c
static UA_StatusCode
pumpTypeConstructor(UA_Server *server,
                    const UA_NodeId *sessionId, void *sessionContext,
                    const UA_NodeId *typeId, void *typeContext,
                    const UA_NodeId *nodeId, void **nodeContext) {
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_USERLAND, "New pump created");

    /* Find the NodeId of the status child variable */
    UA_RelativePathElement rpe;
    UA_RelativePathElement_init(&rpe);
    rpe.referenceTypeId = UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT);
    rpe.isInverse = false;
    rpe.includeSubtypes = false;
    rpe.targetName = UA_QUALIFIEDNAME(1, "Status");

    UA_BrowsePath bp;
    UA_BrowsePath_init(&bp);
    bp.startingNode = *nodeId;
    bp.relativePath.elementsSize = 1;
    bp.relativePath.elements = &rpe;

    UA_BrowsePathResult bpr =
        UA_Server_translateBrowsePathToNodeIds(server, &bp);
    if(bpr.statusCode != UA_STATUSCODE_GOOD ||
       bpr.targetsSize < 1)
        return bpr.statusCode;

    /* Set the status value */
    UA_Boolean status = true;
    UA_Variant value;
    UA_Variant_setScalar(&value, &status, &UA_TYPES[UA_TYPES_BOOLEAN]);
    UA_Server_writeValue(server, bpr.targets[0].targetId.nodeId, value);
    UA_BrowsePathResult_deleteMembers(&bpr);

    /* At this point we could replace the node context .. */

    return UA_STATUSCODE_GOOD;
}

static void
addPumpTypeConstructor(UA_Server *server) {
    UA_NodeTypeLifecycle lifecycle;
    lifecycle.constructor = pumpTypeConstructor;
    lifecycle.destructor = NULL;
    UA_Server_setNodeTypeLifecycle(server, pumpTypeId, lifecycle);
}
```

It follows the main server code, making use of the above definitions.

```
UA_Boolean running = true;
static void stopHandler(int sign) {
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_SERVER, "received ctrl-c");
    running = false;
}

int main(void) {
    signal(SIGINT, stopHandler);
    signal(SIGTERM, stopHandler);

    UA_ServerConfig *config = UA_ServerConfig_new_default();
    UA_Server *server = UA_Server_new(config);

    manuallyDefinePump(server);
    defineObjectTypes(server);
    addPumpObjectInstance(server, "pump2");
    addPumpObjectInstance(server, "pump3");
    addPumpTypeConstructor(server);
    addPumpObjectInstance(server, "pump4");
    addPumpObjectInstance(server, "pump5");

    UA_StatusCode retval = UA_Server_run(server, &running);
    UA_Server_delete(server);
    UA_ServerConfig_delete(config);
    return (int)retval;
}
```

3.7 Adding Methods to Objects

An object in an OPC UA information model may contain methods similar to objects in a programming language. Methods are represented by a MethodNode. Note that several objects may reference the same MethodNode. When an object type is instantiated, a reference to the method is added instead of copying the MethodNode. Therefore, the identifier of the context object is always explicitly stated when a method is called.

The method callback takes as input a custom data pointer attached to the method node, the identifier of the object from which the method is called, and two arrays for the input and output arguments. The input and output arguments are all of type *Variant*. Each variant may in turn contain a (multi-dimensional) array or scalar of any data type.

Constraints for the method arguments are defined in terms of data type, value rank and array dimension (similar to variable definitions). The argument definitions are stored in child VariableNodes of the MethodNode with the respective BrowseNames (0, "InputArguments") and (0, "OutputArguments").

3.7.1 Example: Hello World Method

The method takes a string scalar and returns a string scalar with "Hello " prepended. The type and length of the input arguments is checked internally by the SDK, so that we don't have to verify the arguments in the callback.

```
#include <signal.h>
#include "open62541.h"

static UA_StatusCode
helloWorldMethodCallback(UA_Server *server,
                         const UA_NodeId *sessionId, void *sessionHandle,
                         const UA_NodeId *methodId, void *methodContext,
                         const UA_NodeId *objectId, void *objectContext,
                         size_t inputSize, const UA_Variant *input,
                         size_t outputSize, UA_Variant *output) {
```

```
    UA_String *inputStr = (UA_String*)input->data;
    UA_String tmp = UA_STRING_ALLOC("Hello ");
    if(inputStr->length > 0) {
        tmp.data = (UA_Byte *)UA_realloc(tmp.data, tmp.length + inputStr->length);
        memcpy(&tmp.data[tmp.length], inputStr->data, inputStr->length);
        tmp.length += inputStr->length;
    }
    UA_Variant_setScalarCopy(output, &tmp, &UA_TYPES[UA_TYPES_STRING]);
    UA_String_deleteMembers(&tmp);
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_SERVER, "Hello World was called");
    return UA_STATUSCODE_GOOD;
}

static void
addHellWorldMethod(UA_Server *server) {
    UA_Argument inputArgument;
    UA_Argument_init(&inputArgument);
    inputArgument.description = UA_LOCALIZEDTEXT("en-US", "A String");
    inputArgument.name = UA_STRING("MyInput");
    inputArgument.dataType = UA_TYPES[UA_TYPES_STRING].typeId;
    inputArgument.valueRank = UA_VALUERANK_SCALAR;

    UA_Argument outputArgument;
    UA_Argument_init(&outputArgument);
    outputArgument.description = UA_LOCALIZEDTEXT("en-US", "A String");
    outputArgument.name = UA_STRING("MyOutput");
    outputArgument.dataType = UA_TYPES[UA_TYPES_STRING].typeId;
    outputArgument.valueRank = UA_VALUERANK_SCALAR;

    UA_MethodAttributes helloAttr = UA_MethodAttributes_default;
    helloAttr.description = UA_LOCALIZEDTEXT("en-US","Say `Hello World`");
    helloAttr.displayName = UA_LOCALIZEDTEXT("en-US","Hello World");
    helloAttr.executable = true;
    helloAttr.userExecutable = true;
    UA_Server_addMethodNode(server, UA_NODEID_NUMERIC(1,62541),
                            UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER),
                            UA_NODEID_NUMERIC(0, UA_NS0ID_HASORDEREDCOMPONENT),
                            UA_QUALIFIEDNAME(1, "hello world"),
                            helloAttr, &helloWorldMethodCallback,
                            1, &inputArgument, 1, &outputArgument, NULL, NULL);
}
```

3.7.2 Increase Array Values Method

The method takes an array of 5 integers and a scalar as input. It returns a copy of the array with every entry increased by the scalar.

```
static UA_StatusCode
IncInt32ArrayMethodCallback(UA_Server *server,
                            const UA_NodeId *sessionId, void *sessionContext,
                            const UA_NodeId *methodId, void *methodContext,
                            const UA_NodeId *objectId, void *objectContext,
                            size_t inputSize, const UA_Variant *input,
                            size_t outputSize, UA_Variant *output) {
    UA_Int32 *inputArray = (UA_Int32*)input[0].data;
    UA_Int32 delta = *(UA_Int32*)input[1].data;

    /* Copy the input array */
    UA_StatusCode retval = UA_Variant_setArrayCopy(output, inputArray, 5,
```

```c
                                                    &UA_TYPES[UA_TYPES_INT32]);
    if(retval != UA_STATUSCODE_GOOD)
        return retval;

    /* Increate the elements */
    UA_Int32 *outputArray = (UA_Int32*)output->data;
    for(size_t i = 0; i < input->arrayLength; i++)
        outputArray[i] = inputArray[i] + delta;

    return UA_STATUSCODE_GOOD;
}

static void
addIncInt32ArrayMethod(UA_Server *server) {
    /* Two input arguments */
    UA_Argument inputArguments[2];
    UA_Argument_init(&inputArguments[0]);
    inputArguments[0].description = UA_LOCALIZEDTEXT("en-US", "int32[5] array");
    inputArguments[0].name = UA_STRING("int32 array");
    inputArguments[0].dataType = UA_TYPES[UA_TYPES_INT32].typeId;
    inputArguments[0].valueRank = UA_VALUERANK_ONE_DIMENSION;
    UA_UInt32 pInputDimension = 5;
    inputArguments[0].arrayDimensionsSize = 1;
    inputArguments[0].arrayDimensions = &pInputDimension;

    UA_Argument_init(&inputArguments[1]);
    inputArguments[1].description = UA_LOCALIZEDTEXT("en-US", "int32 delta");
    inputArguments[1].name = UA_STRING("int32 delta");
    inputArguments[1].dataType = UA_TYPES[UA_TYPES_INT32].typeId;
    inputArguments[1].valueRank = UA_VALUERANK_SCALAR;

    /* One output argument */
    UA_Argument outputArgument;
    UA_Argument_init(&outputArgument);
    outputArgument.description = UA_LOCALIZEDTEXT("en-US", "int32[5] array");
    outputArgument.name = UA_STRING("each entry is incremented by the delta");
    outputArgument.dataType = UA_TYPES[UA_TYPES_INT32].typeId;
    outputArgument.valueRank = UA_VALUERANK_ONE_DIMENSION;
    UA_UInt32 pOutputDimension = 5;
    outputArgument.arrayDimensionsSize = 1;
    outputArgument.arrayDimensions = &pOutputDimension;

    /* Add the method node */
    UA_MethodAttributes incAttr = UA_MethodAttributes_default;
    incAttr.description = UA_LOCALIZEDTEXT("en-US", "IncInt32ArrayValues");
    incAttr.displayName = UA_LOCALIZEDTEXT("en-US", "IncInt32ArrayValues");
    incAttr.executable = true;
    incAttr.userExecutable = true;
    UA_Server_addMethodNode(server, UA_NODEID_STRING(1, "IncInt32ArrayValues"),
                            UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER),
                            UA_NODEID_NUMERIC(0, UA_NS0ID_HASCOMPONENT),
                            UA_QUALIFIEDNAME(1, "IncInt32ArrayValues"),
                            incAttr, &IncInt32ArrayMethodCallback,
                            2, inputArguments, 1, &outputArgument,
                            NULL, NULL);
}
```

It follows the main server code, making use of the above definitions.

```c
UA_Boolean running = true;
static void stopHandler(int sign) {
```

3.7. Adding Methods to Objects 27

```
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_SERVER, "received ctrl-c");
    running = false;
}

int main(void) {
    signal(SIGINT, stopHandler);
    signal(SIGTERM, stopHandler);

    UA_ServerConfig *config = UA_ServerConfig_new_default();
    UA_Server *server = UA_Server_new(config);

    addHellWorldMethod(server);
    addIncInt32ArrayMethod(server);

    UA_StatusCode retval = UA_Server_run(server, &running);
    UA_Server_delete(server);
    UA_ServerConfig_delete(config);
    return (int)retval;
}
```

3.8 Building a Simple Client

You should already have a basic server from the previous tutorials. open62541 provides both a server- and clientside API, so creating a client is as easy as creating a server. Copy the following into a file *myClient.c*:

```
#include <stdio.h>
#include "open62541.h"

int main(void) {
    UA_Client *client = UA_Client_new(UA_ClientConfig_default);
    UA_StatusCode retval = UA_Client_connect(client, "opc.tcp://localhost:4840");
    if(retval != UA_STATUSCODE_GOOD) {
        UA_Client_delete(client);
        return (int)retval;
    }

    /* Read the value attribute of the node. UA_Client_readValueAttribute is a
     * wrapper for the raw read service available as UA_Client_Service_read. */
    UA_Variant value; /* Variants can hold scalar values and arrays of any type */
    UA_Variant_init(&value);

    /* NodeId of the variable holding the current time */
    const UA_NodeId nodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_SERVER_SERVERSTATUS_
↪CURRENTTIME);
    retval = UA_Client_readValueAttribute(client, nodeId, &value);

    if(retval == UA_STATUSCODE_GOOD &&
       UA_Variant_hasScalarType(&value, &UA_TYPES[UA_TYPES_DATETIME])) {
        UA_DateTime raw_date = *(UA_DateTime *) value.data;
        UA_DateTimeStruct dts = UA_DateTime_toStruct(raw_date);
        UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_USERLAND, "date is: %u-%u-%u %u:
↪%u:%u.%03u\n",
                    dts.day, dts.month, dts.year, dts.hour, dts.min, dts.sec, dts.
↪milliSec);
    }

    /* Clean up */
    UA_Variant_deleteMembers(&value);
    UA_Client_delete(client); /* Disconnects the client internally */
```

```
    return UA_STATUSCODE_GOOD;
}
```

Compilation is similar to the server example.

```
$ gcc -std=c99 open62541.c myClient.c -o myClient
```

In a MinGW environment, the Winsock library must be added.

```
$ gcc -std=c99 open62541.c myClient.c -lws2_32 -o myClient.exe
```

3.8.1 Further tasks

- Try to connect to some other OPC UA server by changing `opc.tcp://localhost:4840` to an appropriate address (remember that the queried node is contained in any OPC UA server).

- Try to set the value of the variable node (ns=1,i="the.answer") containing an `Int32` from the example server (which is built in *Building a Simple Server*) using "UA_Client_write" function. The example server needs some more modifications, i.e., changing request types. The answer can be found in "examples/exampleClient.c".

Protocol

In this section, we give an overview on the OPC UA binary protocol. We focus on binary since that is what has been implemented in open62541. The TCP-based binary protocol is by far the most common transport layer for OPC UA. The general concepts also translate to HTTP and SOAP-based communication defined in the standard. Communication in OPC UA is best understood by starting with the following key principles:

Request / Response All communication is based on the Request/Response pattern. Only clients can send a request to a server. And servers can only send responses to a request. Usually, the server is hosted on the (physical) device, such as a sensor or a machine tool.

Asynchronous Responses A server does not have to immediately respond to requests and responses may be sent in a different order. This keeps the server responsive when it takes time until a specific request has been processed (e.g. a method call or when reading from a sensor with delay). Furthermore, Subscriptions (aka push-notifications) are implemented via special requests where the response is delayed until a notification is generated.

4.1 Establishing a Connection

A client-server connection in OPC UA consists of three nested levels: The raw connection, a SecureChannel and the Session. For full details, see Part 6 of the OPC UA standard.

Raw Connection The raw connection is created by opening a TCP connection to the corresponding hostname and port and an initial HEL/ACK handshake. The handshake establishes the basic settings of the connection, such as the maximum message length.

SecureChannel SecureChannels are created on top of the raw TCP connection. A SecureChannel is established with an *OpenSecureChannel* request and response message pair. **Attention!** Even though a SecureChannel is mandatory, encryption might still be disabled. The *SecurityMode* of a SecureChannel can be either `None`, `Sign`, or `SignAndEncrypt`. As of version 0.2 of open62541, message signing and encryption is still under ongoing development.

With message signing or encryption enabled, the *OpenSecureChannel* messages are encrypted using an asymmetric encryption algorithm (public-key cryptography)[1]. As part of the *OpenSecureChannel* messages, client and server establish a common secret over an initially unsecure channel. For subsequent messages, the common secret is used for symmetric encryption, which has the advantage of being much faster.

---

[1] This entails that the client and server exchange so-called public keys. The public keys might come with a certificate from a key-signing authority or be verified against an external key repository. But we will not discuss certificate management in detail in this section.

Different *SecurityPolicies* – defined in part 7 of the OPC UA standard – specify the algorithms for asymmetric and symmetric encryption, encryption key lengths, hash functions for message signing, and so on. Example SecurityPolicies are `None` for transmission of cleartext and `Basic256Sha256` which mandates a variant of RSA with SHA256 certificate hashing for asymmetric encryption and AES256 for symmetric encryption.

The possible SecurityPolicies of a server are described with a list of *Endpoints*. An endpoint jointly defines the SecurityMode, SecurityPolicy and means for authenticating a session (discussed in the next section) in order to connect to a certain server. The *GetEndpoints* service returns a list of available endpoints. This service can usually be invoked without a session and from an unencrypted SecureChannel. This allows clients to first discover available endpoints and then use an appropriate SecurityPolicy that might be required to open a session.

Session Sessions are created on top of a SecureChannel. This ensures that users may authenticate without sending their credentials, such as username and password, in cleartext. Currently defined authentication mechanisms are anonymous login, username/password, Kerberos and x509 certificates. The latter requires that the request message is accompanied by a signature to prove that the sender is in possession of the private key with which the certificate was created.

There are two message exchanges required to establish a session: *CreateSession* and *ActicateSession*. The ActivateSession service can be used to switch an existing session to a different SecureChannel. This is important, for example when the connection broke down and the existing session is reused with a new SecureChannel.

4.2 Structure of a protocol message

For the following introduction to the structure of OPC UA protocol messages, consider the example OPC UA binary conversation, recorded and displayed with the Wireshark tool, shown in Fig. 4.1.

The top part of the Wireshark window shows the messages from the conversation in order. The green line contains the applied filter. Here, we want to see the OPC UA protocol messages only. The first messages (from TCP packets 49 to 56) show the client opening an unencrypted SecureChannel and retrieving the server's endpoints. Then, starting with packet 63, a new connection and SecureChannel are created in conformance with one of the endpoints. On top of this SecureChannel, the client can then create and activate a session. The following *ReadRequest* message is selected and covered in more detail in the bottom windows.

The bottom left window shows the structure of the selected *ReadRequest* message. The purpose of the message is invoking the *Read service*. The message is structured into a header and a message body. Note that we do not consider encryption or signing of messages here.

Message Header As stated before, OPC UA defines an asynchronous protocol. So responses may be out of order. The message header contains some basic information, such as the length of the message, as well as necessary information to relate messages to a SecureChannel and each request to the corresponding response. "Chunking" refers to the splitting and reassembling of messages that are longer than the maximum network packet size.

Message Body Every OPC UA *service* has a signature in the form of a request and response data structure. These are defined according to the OPC UA protocol *type system*. See especially the *auto-generated type definitions* for the data types corresponding to service requests and responses. The message body begins with the identifier of the following data type. Then, the main payload of the message follows.

The bottom right window shows the binary payload of the selected *ReadRequest* message. The message header is highlighted in light-grey. The message body in blue highlighting shows the encoded *ReadRequest* data structure.

Fig. 4.1: OPC UA conversation displayed in Wireshark

CHAPTER 5

Data Types

The OPC UA protocol defines 25 builtin data types and three ways of combining them into higher-order types: arrays, structures and unions. In open62541, only the builtin data types are defined manually. All other data types are generated from standard XML definitions. Their exact definitions can be looked up at https://opcfoundation. org/UA/schemas/Opc.Ua.Types.bsd.xml.

For users that are new to open62541, take a look at the *tutorial for working with data types* before diving into the implementation details.

5.1 Builtin Types

5.1.1 Boolean

A two-state logical value (true or false).

```
typedef bool UA_Boolean;
#define UA_TRUE true
#define UA_FALSE false
```

5.1.2 SByte

An integer value between -128 and 127.

```
typedef int8_t UA_SByte;
#define UA_SBYTE_MIN (-128)
#define UA_SBYTE_MAX 127
```

5.1.3 Byte

An integer value between 0 and 255.

```
typedef uint8_t UA_Byte;
#define UA_BYTE_MIN 0
#define UA_BYTE_MAX 255
```

5.1.4 Int16

An integer value between -32 768 and 32 767.

```
typedef int16_t UA_Int16;
#define UA_INT16_MIN (-32768)
#define UA_INT16_MAX 32767
```

5.1.5 UInt16

An integer value between 0 and 65 535.

```
typedef uint16_t UA_UInt16;
#define UA_UINT16_MIN 0
#define UA_UINT16_MAX 65535
```

5.1.6 Int32

An integer value between -2 147 483 648 and 2 147 483 647.

```
typedef int32_t UA_Int32;
#define UA_INT32_MIN (-2147483648)
#define UA_INT32_MAX 2147483647
```

5.1.7 UInt32

An integer value between 0 and 4 294 967 295.

```
typedef uint32_t UA_UInt32;
#define UA_UINT32_MIN 0
#define UA_UINT32_MAX 4294967295
```

5.1.8 Int64

An integer value between -9 223 372 036 854 775 808 and 9 223 372 036 854 775 807.

```
typedef int64_t UA_Int64;
#define UA_INT64_MIN ((int64_t)-9223372036854775808)
#define UA_INT64_MAX (int64_t)9223372036854775807
```

5.1.9 UInt64

An integer value between 0 and 18 446 744 073 709 551 615.

```
typedef uint64_t UA_UInt64;
#define UA_UINT64_MIN (int64_t)0
#define UA_UINT64_MAX (int64_t)18446744073709551615
```

5.1.10 Float

An IEEE single precision (32 bit) floating point value.

```
typedef float UA_Float;
```

5.1.11 Double

An IEEE double precision (64 bit) floating point value.

```
typedef double UA_Double;
```

5.1.12 StatusCode

A numeric identifier for a error or condition that is associated with a value or an operation. See the section *StatusCodes* for the meaning of a specific code.

```
typedef uint32_t UA_StatusCode;

/* Returns the human-readable name of the StatusCode. If no matching StatusCode
 * is found, a default string for "Unknown" is returned. This feature might be
 * disabled to create a smaller binary with the
 * UA_ENABLE_STATUSCODE_DESCRIPTIONS build-flag. Then the function returns an
 * empty string for every StatusCode. */
const char *
UA_StatusCode_name(UA_StatusCode code);
```

5.1.13 String

A sequence of Unicode characters. Strings are just an array of UA_Byte.

```
typedef struct {
    size_t length; /* The length of the string */
    UA_Byte *data; /* The content (not null-terminated) */
} UA_String;

/* Copies the content on the heap. Returns a null-string when alloc fails */
UA_String UA_String_fromChars(char const src[]);

UA_Boolean UA_String_equal(const UA_String *s1, const UA_String *s2);

extern const UA_String UA_STRING_NULL;
```

UA_STRING returns a string pointing to the original char-array. UA_STRING_ALLOC is shorthand for UA_String_fromChars and makes a copy of the char-array.

```
static UA_INLINE UA_String
UA_STRING(char *chars) {
    UA_String str; str.length = strlen(chars);
    str.data = (UA_Byte*)chars; return str;
}

#define UA_STRING_ALLOC(CHARS) UA_String_fromChars(CHARS)

/* Define strings at compile time (in ROM) */
#define UA_STRING_STATIC(CHARS) {sizeof(CHARS)-1, (UA_Byte*)CHARS}
```

5.1.14 DateTime

An instance in time. A DateTime value is encoded as a 64-bit signed integer which represents the number of 100 nanosecond intervals since January 1, 1601 (UTC).

The methods providing an interface to the system clock are provided by a "plugin" that is statically linked with the library.

```
typedef int64_t UA_DateTime;

/* Multiples to convert durations to DateTime */
#define UA_DATETIME_USEC 10LL
#define UA_DATETIME_MSEC (UA_DATETIME_USEC * 1000LL)
#define UA_DATETIME_SEC (UA_DATETIME_MSEC * 1000LL)

/* The current time in UTC time */
UA_DateTime UA_DateTime_now(void);

/* Offset between local time and UTC time */
UA_Int64 UA_DateTime_localTimeUtcOffset(void);

/* CPU clock invariant to system time changes. Use only to measure durations,
 * not absolute time. */
UA_DateTime UA_DateTime_nowMonotonic(void);

/* Represents a Datetime as a structure */
typedef struct UA_DateTimeStruct {
    UA_UInt16 nanoSec;
    UA_UInt16 microSec;
    UA_UInt16 milliSec;
    UA_UInt16 sec;
    UA_UInt16 min;
    UA_UInt16 hour;
    UA_UInt16 day;
    UA_UInt16 month;
    UA_UInt16 year;
} UA_DateTimeStruct;

UA_DateTimeStruct UA_DateTime_toStruct(UA_DateTime t);

/* The C99 standard (7.23.1) says: "The range and precision of times
 * representable in clock_t and time_t are implementation-defined." On most
 * systems, time_t is a 4 or 8 byte integer counting seconds since the UTC Unix
 * epoch. The following methods are used for conversion. */

/* Datetime of 1 Jan 1970 00:00 */
#define UA_DATETIME_UNIX_EPOCH (11644473600LL * UA_DATETIME_SEC)

static UA_INLINE UA_Int64
UA_DateTime_toUnixTime(UA_DateTime date) {
    return (date - UA_DATETIME_UNIX_EPOCH) / UA_DATETIME_SEC;
}

static UA_INLINE UA_DateTime
UA_DateTime_fromUnixTime(UA_Int64 unixDate) {
    return (unixDate * UA_DATETIME_SEC) + UA_DATETIME_UNIX_EPOCH;
}
```

5.1.15 Guid

A 16 byte value that can be used as a globally unique identifier.

```c
typedef struct {
    UA_UInt32 data1;
    UA_UInt16 data2;
    UA_UInt16 data3;
    UA_Byte   data4[8];
} UA_Guid;

UA_Boolean UA_Guid_equal(const UA_Guid *g1, const UA_Guid *g2);

extern const UA_Guid UA_GUID_NULL;
```

5.1.16 ByteString

A sequence of octets.

```c
typedef UA_String UA_ByteString;

static UA_INLINE UA_Boolean
UA_ByteString_equal(const UA_ByteString *string1,
                    const UA_ByteString *string2) {
    return UA_String_equal((const UA_String*)string1,
                           (const UA_String*)string2);
}

/* Allocates memory of size length for the bytestring.
 * The content is not set to zero. */
UA_StatusCode
UA_ByteString_allocBuffer(UA_ByteString *bs, size_t length);

extern const UA_ByteString UA_BYTESTRING_NULL;

static UA_INLINE UA_ByteString
UA_BYTESTRING(char *chars) {
    UA_ByteString str; str.length = strlen(chars);
    str.data = (UA_Byte*)chars; return str;
}

static UA_INLINE UA_ByteString
UA_BYTESTRING_ALLOC(const char *chars) {
    UA_String str = UA_String_fromChars(chars); UA_ByteString bstr;
    bstr.length = str.length; bstr.data = str.data; return bstr;
}
```

5.1.17 XmlElement

An XML element.

```c
typedef UA_String UA_XmlElement;
```

5.1.18 NodeId

An identifier for a node in the address space of an OPC UA Server.

```c
enum UA_NodeIdType {
    UA_NODEIDTYPE_NUMERIC   = 0, /* In the binary encoding, this can also
                                    become 1 or 2 (2byte and 4byte encoding of
                                    small numeric nodeids) */
```

(continues on next page)

```
    UA_NODEIDTYPE_STRING    = 3,
    UA_NODEIDTYPE_GUID      = 4,
    UA_NODEIDTYPE_BYTESTRING = 5
};

typedef struct {
    UA_UInt16 namespaceIndex;
    enum UA_NodeIdType identifierType;
    union {
        UA_UInt32    numeric;
        UA_String    string;
        UA_Guid      guid;
        UA_ByteString byteString;
    } identifier;
} UA_NodeId;

extern const UA_NodeId UA_NODEID_NULL;

UA_Boolean UA_NodeId_isNull(const UA_NodeId *p);

UA_Boolean UA_NodeId_equal(const UA_NodeId *n1, const UA_NodeId *n2);

/* Returns a non-cryptographic hash for the NodeId */
UA_UInt32 UA_NodeId_hash(const UA_NodeId *n);
```

The following functions are shorthand for creating NodeIds.

```
static UA_INLINE UA_NodeId
UA_NODEID_NUMERIC(UA_UInt16 nsIndex, UA_UInt32 identifier) {
    UA_NodeId id; id.namespaceIndex = nsIndex;
    id.identifierType = UA_NODEIDTYPE_NUMERIC;
    id.identifier.numeric = identifier; return id;
}

static UA_INLINE UA_NodeId
UA_NODEID_STRING(UA_UInt16 nsIndex, char *chars) {
    UA_NodeId id; id.namespaceIndex = nsIndex;
    id.identifierType = UA_NODEIDTYPE_STRING;
    id.identifier.string = UA_STRING(chars); return id;
}

static UA_INLINE UA_NodeId
UA_NODEID_STRING_ALLOC(UA_UInt16 nsIndex, const char *chars) {
    UA_NodeId id; id.namespaceIndex = nsIndex;
    id.identifierType = UA_NODEIDTYPE_STRING;
    id.identifier.string = UA_STRING_ALLOC(chars); return id;
}

static UA_INLINE UA_NodeId
UA_NODEID_GUID(UA_UInt16 nsIndex, UA_Guid guid) {
    UA_NodeId id; id.namespaceIndex = nsIndex;
    id.identifierType = UA_NODEIDTYPE_GUID;
    id.identifier.guid = guid; return id;
}

static UA_INLINE UA_NodeId
UA_NODEID_BYTESTRING(UA_UInt16 nsIndex, char *chars) {
    UA_NodeId id; id.namespaceIndex = nsIndex;
    id.identifierType = UA_NODEIDTYPE_BYTESTRING;
    id.identifier.byteString = UA_BYTESTRING(chars); return id;
}
```

```
static UA_INLINE UA_NodeId
UA_NODEID_BYTESTRING_ALLOC(UA_UInt16 nsIndex, const char *chars) {
    UA_NodeId id; id.namespaceIndex = nsIndex;
    id.identifierType = UA_NODEIDTYPE_BYTESTRING;
    id.identifier.byteString = UA_BYTESTRING_ALLOC(chars); return id;
}
```

5.1.19 ExpandedNodeId

A NodeId that allows the namespace URI to be specified instead of an index.

```
typedef struct {
    UA_NodeId nodeId;
    UA_String namespaceUri;
    UA_UInt32 serverIndex;
} UA_ExpandedNodeId;

UA_Boolean UA_ExpandedNodeId_equal(const UA_ExpandedNodeId *n1,
                                   const UA_ExpandedNodeId *n2);

extern const UA_ExpandedNodeId UA_EXPANDEDNODEID_NULL;
```

The following functions are shorthand for creating ExpandedNodeIds.

```
static UA_INLINE UA_ExpandedNodeId
UA_EXPANDEDNODEID_NUMERIC(UA_UInt16 nsIndex, UA_UInt32 identifier) {
    UA_ExpandedNodeId id; id.nodeId = UA_NODEID_NUMERIC(nsIndex, identifier);
    id.serverIndex = 0; id.namespaceUri = UA_STRING_NULL; return id;
}

static UA_INLINE UA_ExpandedNodeId
UA_EXPANDEDNODEID_STRING(UA_UInt16 nsIndex, char *chars) {
    UA_ExpandedNodeId id; id.nodeId = UA_NODEID_STRING(nsIndex, chars);
    id.serverIndex = 0; id.namespaceUri = UA_STRING_NULL; return id;
}

static UA_INLINE UA_ExpandedNodeId
UA_EXPANDEDNODEID_STRING_ALLOC(UA_UInt16 nsIndex, const char *chars) {
    UA_ExpandedNodeId id; id.nodeId = UA_NODEID_STRING_ALLOC(nsIndex, chars);
    id.serverIndex = 0; id.namespaceUri = UA_STRING_NULL; return id;
}

static UA_INLINE UA_ExpandedNodeId
UA_EXPANDEDNODEID_STRING_GUID(UA_UInt16 nsIndex, UA_Guid guid) {
    UA_ExpandedNodeId id; id.nodeId = UA_NODEID_GUID(nsIndex, guid);
    id.serverIndex = 0; id.namespaceUri = UA_STRING_NULL; return id;
}

static UA_INLINE UA_ExpandedNodeId
UA_EXPANDEDNODEID_BYTESTRING(UA_UInt16 nsIndex, char *chars) {
    UA_ExpandedNodeId id; id.nodeId = UA_NODEID_BYTESTRING(nsIndex, chars);
    id.serverIndex = 0; id.namespaceUri = UA_STRING_NULL; return id;
}

static UA_INLINE UA_ExpandedNodeId
UA_EXPANDEDNODEID_BYTESTRING_ALLOC(UA_UInt16 nsIndex, const char *chars) {
    UA_ExpandedNodeId id; id.nodeId = UA_NODEID_BYTESTRING_ALLOC(nsIndex, chars);
    id.serverIndex = 0; id.namespaceUri = UA_STRING_NULL; return id;
}
```

5.1.20 QualifiedName

A name qualified by a namespace.

```
typedef struct {
    UA_UInt16 namespaceIndex;
    UA_String name;
} UA_QualifiedName;

static UA_INLINE UA_Boolean
UA_QualifiedName_isNull(const UA_QualifiedName *q) {
    return (q->namespaceIndex == 0 && q->name.length == 0);
}

static UA_INLINE UA_QualifiedName
UA_QUALIFIEDNAME(UA_UInt16 nsIndex, char *chars) {
    UA_QualifiedName qn; qn.namespaceIndex = nsIndex;
    qn.name = UA_STRING(chars); return qn;
}

static UA_INLINE UA_QualifiedName
UA_QUALIFIEDNAME_ALLOC(UA_UInt16 nsIndex, const char *chars) {
    UA_QualifiedName qn; qn.namespaceIndex = nsIndex;
    qn.name = UA_STRING_ALLOC(chars); return qn;
}

UA_Boolean
UA_QualifiedName_equal(const UA_QualifiedName *qn1,
                       const UA_QualifiedName *qn2);
```

5.1.21 LocalizedText

Human readable text with an optional locale identifier.

```
typedef struct {
    UA_String locale;
    UA_String text;
} UA_LocalizedText;

static UA_INLINE UA_LocalizedText
UA_LOCALIZEDTEXT(char *locale, char *text) {
    UA_LocalizedText lt; lt.locale = UA_STRING(locale);
    lt.text = UA_STRING(text); return lt;
}

static UA_INLINE UA_LocalizedText
UA_LOCALIZEDTEXT_ALLOC(const char *locale, const char *text) {
    UA_LocalizedText lt; lt.locale = UA_STRING_ALLOC(locale);
    lt.text = UA_STRING_ALLOC(text); return lt;
}
```

5.1.22 NumericRange

NumericRanges are used to indicate subsets of a (multidimensional) array. They no official data type in the OPC UA standard and are transmitted only with a string encoding, such as "1:2,0:3,5". The colon separates min/max index and the comma separates dimensions. A single value indicates a range with a single element (min==max).

```
typedef struct {
    UA_UInt32 min;
```

```
    UA_UInt32 max;
} UA_NumericRangeDimension;

typedef struct {
    size_t dimensionsSize;
    UA_NumericRangeDimension *dimensions;
} UA_NumericRange;
```

5.1.23 Variant

Variants may contain values of any type together with a description of the content. See the section on *Generic Type Handling* on how types are described. The standard mandates that variants contain built-in data types only. If the value is not of a builtin type, it is wrapped into an *ExtensionObject*. open62541 hides this wrapping transparently in the encoding layer. If the data type is unknown to the receiver, the variant contains the original ExtensionObject in binary or XML encoding.

Variants may contain a scalar value or an array. For details on the handling of arrays, see the section on *Array handling*. Array variants can have an additional dimensionality (matrix, 3-tensor, . . .) defined in an array of dimension lengths. The actual values are kept in an array of dimensions one. For users who work with higher-dimensions arrays directly, keep in mind that dimensions of higher rank are serialized first (the highest rank dimension has stride 1 and elements follow each other directly). Usually it is simplest to interact with higher-dimensional arrays via UA_NumericRange descriptions (see *Array handling*).

To differentiate between scalar / array variants, the following definition is used. UA_Variant_isScalar provides simplified access to these checks.

- arrayLength == 0 && data == NULL: undefined array of length -1

- arrayLength == 0 && data == UA_EMPTY_ARRAY_SENTINEL: array of length 0

- arrayLength == 0 && data > UA_EMPTY_ARRAY_SENTINEL: scalar value

- arrayLength > 0: array of the given length

Variants can also be *empty*. Then, the pointer to the type description is NULL.

```
/* Forward declaration. See the section on Generic Type Handling */
struct UA_DataType;
typedef struct UA_DataType UA_DataType;

#define UA_EMPTY_ARRAY_SENTINEL ((void*)0x01)

typedef enum {
    UA_VARIANT_DATA,          /* The data has the same lifecycle as the
                                 variant */
    UA_VARIANT_DATA_NODELETE /* The data is "borrowed" by the variant and
                                shall not be deleted at the end of the
                                variant's lifecycle. */
} UA_VariantStorageType;

typedef struct {
    const UA_DataType *type;      /* The data type description */
    UA_VariantStorageType storageType;
    size_t arrayLength;           /* The number of elements in the data array */
    void *data;                   /* Points to the scalar or array data */
    size_t arrayDimensionsSize;   /* The number of dimensions */
    UA_UInt32 *arrayDimensions;   /* The length of each dimension */
} UA_Variant;

/* Returns true if the variant has no value defined (contains neither an array
 * nor a scalar value).
```

```
 *
 * @param v The variant
 * @return Is the variant empty */
static UA_INLINE UA_Boolean
UA_Variant_isEmpty(const UA_Variant *v) {
    return v->type == NULL;
}

/* Returns true if the variant contains a scalar value. Note that empty variants
 * contain an array of length -1 (undefined).
 *
 * @param v The variant
 * @return Does the variant contain a scalar value */
static UA_INLINE UA_Boolean
UA_Variant_isScalar(const UA_Variant *v) {
    return (v->arrayLength == 0 && v->data > UA_EMPTY_ARRAY_SENTINEL);
}

/* Returns true if the variant contains a scalar value of the given type.
 *
 * @param v The variant
 * @param type The data type
 * @return Does the variant contain a scalar value of the given type */
static UA_INLINE UA_Boolean
UA_Variant_hasScalarType(const UA_Variant *v, const UA_DataType *type) {
    return UA_Variant_isScalar(v) && type == v->type;
}

/* Returns true if the variant contains an array of the given type.
 *
 * @param v The variant
 * @param type The data type
 * @return Does the variant contain an array of the given type */
static UA_INLINE UA_Boolean
UA_Variant_hasArrayType(const UA_Variant *v, const UA_DataType *type) {
    return (!UA_Variant_isScalar(v)) && type == v->type;
}

/* Set the variant to a scalar value that already resides in memory. The value
 * takes on the lifecycle of the variant and is deleted with it.
 *
 * @param v The variant
 * @param p A pointer to the value data
 * @param type The datatype of the value in question */
void
UA_Variant_setScalar(UA_Variant *v, void *p,
                     const UA_DataType *type);

/* Set the variant to a scalar value that is copied from an existing variable.
 * @param v The variant
 * @param p A pointer to the value data
 * @param type The datatype of the value
 * @return Indicates whether the operation succeeded or returns an error code */
UA_StatusCode
UA_Variant_setScalarCopy(UA_Variant *v, const void *p,
                         const UA_DataType *type);

/* Set the variant to an array that already resides in memory. The array takes
 * on the lifecycle of the variant and is deleted with it.
 *
 * @param v The variant
```

```
 * @param array A pointer to the array data
 * @param arraySize The size of the array
 * @param type The datatype of the array */
void
UA_Variant_setArray(UA_Variant *v, void *array,
                    size_t arraySize, const UA_DataType *type);

/* Set the variant to an array that is copied from an existing array.
 *
 * @param v The variant
 * @param array A pointer to the array data
 * @param arraySize The size of the array
 * @param type The datatype of the array
 * @return Indicates whether the operation succeeded or returns an error code */
UA_StatusCode
UA_Variant_setArrayCopy(UA_Variant *v, const void *array,
                        size_t arraySize, const UA_DataType *type);

/* Copy the variant, but use only a subset of the (multidimensional) array into
 * a variant. Returns an error code if the variant is not an array or if the
 * indicated range does not fit.
 *
 * @param src The source variant
 * @param dst The target variant
 * @param range The range of the copied data
 * @return Returns UA_STATUSCODE_GOOD or an error code */
UA_StatusCode
UA_Variant_copyRange(const UA_Variant *src, UA_Variant *dst,
                     const UA_NumericRange range);

/* Insert a range of data into an existing variant. The data array can't be
 * reused afterwards if it contains types without a fixed size (e.g. strings)
 * since the members are moved into the variant and take on its lifecycle.
 *
 * @param v The variant
 * @param dataArray The data array. The type must match the variant
 * @param dataArraySize The length of the data array. This is checked to match
 *        the range size.
 * @param range The range of where the new data is inserted
 * @return Returns UA_STATUSCODE_GOOD or an error code */
UA_StatusCode
UA_Variant_setRange(UA_Variant *v, void *array,
                    size_t arraySize, const UA_NumericRange range);

/* Deep-copy a range of data into an existing variant.
 *
 * @param v The variant
 * @param dataArray The data array. The type must match the variant
 * @param dataArraySize The length of the data array. This is checked to match
 *        the range size.
 * @param range The range of where the new data is inserted
 * @return Returns UA_STATUSCODE_GOOD or an error code */
UA_StatusCode
UA_Variant_setRangeCopy(UA_Variant *v, const void *array,
                        size_t arraySize, const UA_NumericRange range);
```

5.1.24 ExtensionObject

ExtensionObjects may contain scalars of any data type. Even those that are unknown to the receiver. See the section on *Generic Type Handling* on how types are described. If the received data type is unknown, the encoded

string and target NodeId is stored instead of the decoded value.

```
typedef enum {
    UA_EXTENSIONOBJECT_ENCODED_NOBODY     = 0,
    UA_EXTENSIONOBJECT_ENCODED_BYTESTRING = 1,
    UA_EXTENSIONOBJECT_ENCODED_XML        = 2,
    UA_EXTENSIONOBJECT_DECODED            = 3,
    UA_EXTENSIONOBJECT_DECODED_NODELETE   = 4 /* Don't delete the content
                                                 together with the
                                                 ExtensionObject */
} UA_ExtensionObjectEncoding;

typedef struct {
    UA_ExtensionObjectEncoding encoding;
    union {
        struct {
            UA_NodeId typeId;   /* The nodeid of the datatype */
            UA_ByteString body; /* The bytestring of the encoded data */
        } encoded;
        struct {
            const UA_DataType *type;
            void *data;
        } decoded;
    } content;
} UA_ExtensionObject;
```

5.1.25 DataValue

A data value with an associated status code and timestamps.

```
typedef struct {
    UA_Boolean     hasValue             : 1;
    UA_Boolean     hasStatus            : 1;
    UA_Boolean     hasSourceTimestamp   : 1;
    UA_Boolean     hasServerTimestamp   : 1;
    UA_Boolean     hasSourcePicoseconds : 1;
    UA_Boolean     hasServerPicoseconds : 1;
    UA_Variant     value;
    UA_StatusCode  status;
    UA_DateTime    sourceTimestamp;
    UA_UInt16      sourcePicoseconds;
    UA_DateTime    serverTimestamp;
    UA_UInt16      serverPicoseconds;
} UA_DataValue;
```

5.1.26 DiagnosticInfo

A structure that contains detailed error and diagnostic information associated with a StatusCode.

```
typedef struct UA_DiagnosticInfo {
    UA_Boolean     hasSymbolicId        : 1;
    UA_Boolean     hasNamespaceUri      : 1;
    UA_Boolean     hasLocalizedText     : 1;
    UA_Boolean     hasLocale            : 1;
    UA_Boolean     hasAdditionalInfo    : 1;
    UA_Boolean     hasInnerStatusCode   : 1;
    UA_Boolean     hasInnerDiagnosticInfo : 1;
    UA_Int32       symbolicId;
    UA_Int32       namespaceUri;
```

(continues on next page)

```
    UA_Int32      localizedText;
    UA_Int32      locale;
    UA_String     additionalInfo;
    UA_StatusCode innerStatusCode;
    struct UA_DiagnosticInfo *innerDiagnosticInfo;
} UA_DiagnosticInfo;
```

5.2 Generic Type Handling

All information about a (builtin/structured) data type is stored in a `UA_DataType`. The array `UA_TYPES` contains the description of all standard-defined types. This type description is used for the following generic operations that work on all types:

- `void T_init(T *ptr)`: Initialize the data type. This is synonymous with zeroing out the memory, i.e. `memset(ptr, 0, sizeof(T))`.

- `T* T_new()`: Allocate and return the memory for the data type. The value is already initialized.

- `UA_StatusCode T_copy(const T *src, T *dst)`: Copy the content of the data type. Returns `UA_STATUSCODE_GOOD` or `UA_STATUSCODE_BADOUTOFMEMORY`.

- `void T_deleteMembers(T *ptr)`: Delete the dynamically allocated content of the data type and perform a `T_init` to reset the type.

- `void T_delete(T *ptr)`: Delete the content of the data type and the memory for the data type itself.

Specializations, such as `UA_Int32_new()` are derived from the generic type operations as static inline functions.

```
typedef struct {
#ifdef UA_ENABLE_TYPENAMES
    const char *memberName;
#endif
    UA_UInt16 memberTypeIndex;    /* Index of the member in the array of data
                                     types */
    UA_Byte   padding;            /* How much padding is there before this
                                     member element? For arrays this is the
                                     padding before the size_t length member.
                                     (No padding between size_t and the
                                     following ptr.) */
    UA_Boolean namespaceZero : 1; /* The type of the member is defined in
                                     namespace zero. In this implementation,
                                     types from custom namespace may contain
                                     members from the same namespace or
                                     namespace zero only.*/
    UA_Boolean isArray      : 1; /* The member is an array */
} UA_DataTypeMember;

struct UA_DataType {
#ifdef UA_ENABLE_TYPENAMES
    const char *typeName;
#endif
    UA_NodeId  typeId;           /* The nodeid of the type */
    UA_UInt16  memSize;          /* Size of the struct in memory */
    UA_UInt16  typeIndex;        /* Index of the type in the datatypetable */
    UA_Byte    membersSize;      /* How many members does the type have? */
    UA_Boolean builtin     : 1; /* The type is "builtin" and has dedicated de-
                                    and encoding functions */
    UA_Boolean pointerFree : 1; /* The type (and its members) contains no
                                    pointers that need to be freed */
```

```
    UA_Boolean overlayable  : 1; /* The type has the identical memory layout in
                                     memory and on the binary stream. */
    UA_UInt16  binaryEncodingId; /* NodeId of datatype when encoded as binary */
    //UA_UInt16  xmlEncodingId;  /* NodeId of datatype when encoded as XML */
    UA_DataTypeMember *members;
};


/* Test if the data type is a numeric builtin data type. This includes Boolean,
 * integers and floating point numbers. Not included are DateTime and
 * StatusCode. */
UA_Boolean
isDataTypeNumeric(const UA_DataType *type);

/* The following is used to exclude type names in the definition of UA_DataType
 * structures if the feature is disabled. */
#ifdef UA_ENABLE_TYPENAMES
# define UA_TYPENAME(name) name,
#else
# define UA_TYPENAME(name)
#endif
```

Builtin data types can be accessed as UA_TYPES[UA_TYPES_XXX], where XXX is the name of the data type. If only the NodeId of a type is known, use the following method to retrieve the data type description.

```
/* Returns the data type description for the type's identifier or NULL if no
 * matching data type was found. */
const UA_DataType *
UA_findDataType(const UA_NodeId *typeId);
```

The following functions are used for generic handling of data types.

```
/* Allocates and initializes a variable of type dataType
 *
 * @param type The datatype description
 * @return Returns the memory location of the variable or NULL if no
 *         memory could be allocated */
void * UA_new(const UA_DataType *type);

/* Initializes a variable to default values
 *
 * @param p The memory location of the variable
 * @param type The datatype description */
static UA_INLINE void
UA_init(void *p, const UA_DataType *type) {
    memset(p, 0, type->memSize);
}

/* Copies the content of two variables. If copying fails (e.g. because no memory
 * was available for an array), then dst is emptied and initialized to prevent
 * memory leaks.
 *
 * @param src The memory location of the source variable
 * @param dst The memory location of the destination variable
 * @param type The datatype description
 * @return Indicates whether the operation succeeded or returns an error code */
UA_StatusCode
UA_copy(const void *src, void *dst, const UA_DataType *type);

/* Deletes the dynamically allocated content of a variable (e.g. resets all
 * arrays to undefined arrays). Afterwards, the variable can be safely deleted
 * without causing memory leaks. But the variable is not initialized and may
```

Chapter 5. Data Types

```
 * contain old data that is not memory-relevant.
 *
 * @param p The memory location of the variable
 * @param type The datatype description of the variable */
void UA_deleteMembers(void *p, const UA_DataType *type);

/* Frees a variable and all of its content.
 *
 * @param p The memory location of the variable
 * @param type The datatype description of the variable */
void UA_delete(void *p, const UA_DataType *type);
```

5.3 Array handling

In OPC UA, arrays can have a length of zero or more with the usual meaning. In addition, arrays can be undefined. Then, they don't even have a length. In the binary encoding, this is indicated by an array of length -1.

In open62541 however, we use `size_t` for array lengths. An undefined array has length 0 and the data pointer is `NULL`. An array of length 0 also has length 0 but a data pointer `UA_EMPTY_ARRAY_SENTINEL`.

```
/* Allocates and initializes an array of variables of a specific type
 *
 * @param size The requested array length
 * @param type The datatype description
 * @return Returns the memory location of the variable or NULL if no memory
 *         could be allocated */
void * UA_Array_new(size_t size, const UA_DataType *type);

/* Allocates and copies an array
 *
 * @param src The memory location of the source array
 * @param size The size of the array
 * @param dst The location of the pointer to the new array
 * @param type The datatype of the array members
 * @return Returns UA_STATUSCODE_GOOD or UA_STATUSCODE_BADOUTOFMEMORY */
UA_StatusCode
UA_Array_copy(const void *src, size_t size, void **dst,
              const UA_DataType *type);

/* Deletes an array.
 *
 * @param p The memory location of the array
 * @param size The size of the array
 * @param type The datatype of the array members */
void UA_Array_delete(void *p, size_t size, const UA_DataType *type);
```

5.4 Random Number Generator

If UA_ENABLE_MULTITHREADING is defined, then the seed is stored in thread local storage. The seed is initialized for every thread in the server/client.

```
void UA_random_seed(UA_UInt64 seed);
UA_UInt32 UA_UInt32_random(void); /* no cryptographic entropy */
UA_Guid UA_Guid_random(void);     /* no cryptographic entropy */
```

5.5 Generated Data Type Definitions

The following data types were auto-generated from a definition in XML format.

Every type is assigned an index in an array containing the type descriptions. These descriptions are used during type handling (copying, deletion, binary encoding, . . .).

```
#define UA_TYPES_COUNT 199
extern const UA_DataType UA_TYPES[UA_TYPES_COUNT];
```

5.5.1 Boolean

```
#define UA_TYPES_BOOLEAN 0
```

5.5.2 SByte

```
#define UA_TYPES_SBYTE 1
```

5.5.3 Byte

```
#define UA_TYPES_BYTE 2
```

5.5.4 Int16

```
#define UA_TYPES_INT16 3
```

5.5.5 UInt16

```
#define UA_TYPES_UINT16 4
```

5.5.6 Int32

```
#define UA_TYPES_INT32 5
```

5.5.7 UInt32

```
#define UA_TYPES_UINT32 6
```

5.5.8 Int64

```
#define UA_TYPES_INT64 7
```

5.5.9 UInt64

```
#define UA_TYPES_UINT64 8
```

5.5.10 Float

```
#define UA_TYPES_FLOAT 9
```

5.5.11 Double

```
#define UA_TYPES_DOUBLE 10
```

5.5.12 String

```
#define UA_TYPES_STRING 11
```

5.5.13 DateTime

```
#define UA_TYPES_DATETIME 12
```

5.5.14 Guid

```
#define UA_TYPES_GUID 13
```

5.5.15 ByteString

```
#define UA_TYPES_BYTESTRING 14
```

5.5.16 XmlElement

```
#define UA_TYPES_XMLELEMENT 15
```

5.5.17 NodeId

```
#define UA_TYPES_NODEID 16
```

5.5.18 ExpandedNodeId

```
#define UA_TYPES_EXPANDEDNODEID 17
```

5.5.19 StatusCode

```
#define UA_TYPES_STATUSCODE 18
```

5.5.20 QualifiedName

```
#define UA_TYPES_QUALIFIEDNAME 19
```

5.5.21 LocalizedText

```
#define UA_TYPES_LOCALIZEDTEXT 20
```

5.5.22 ExtensionObject

```
#define UA_TYPES_EXTENSIONOBJECT 21
```

5.5.23 DataValue

```
#define UA_TYPES_DATAVALUE 22
```

5.5.24 Variant

```
#define UA_TYPES_VARIANT 23
```

5.5.25 DiagnosticInfo

```
#define UA_TYPES_DIAGNOSTICINFO 24
```

5.5.26 SignedSoftwareCertificate

A software certificate with a digital signature.

```
typedef struct {
    UA_ByteString certificateData;
    UA_ByteString signature;
} UA_SignedSoftwareCertificate;

#define UA_TYPES_SIGNEDSOFTWARECERTIFICATE 25
```

5.5.27 SemanticChangeStructureDataType

```
typedef struct {
    UA_NodeId affected;
    UA_NodeId affectedType;
} UA_SemanticChangeStructureDataType;

#define UA_TYPES_SEMANTICCHANGESTRUCTUREDATATYPE 26
```

5.5.28 StatusChangeNotification

```
typedef struct {
    UA_StatusCode status;
    UA_DiagnosticInfo diagnosticInfo;
} UA_StatusChangeNotification;

#define UA_TYPES_STATUSCHANGENOTIFICATION 27
```

5.5.29 BrowsePathTarget

The target of the translated path.

```
typedef struct {
    UA_ExpandedNodeId targetId;
    UA_UInt32 remainingPathIndex;
} UA_BrowsePathTarget;

#define UA_TYPES_BROWSEPATHTARGET 28
```

5.5.30 ViewAttributes

The attributes for a view node.

```
typedef struct {
    UA_UInt32 specifiedAttributes;
    UA_LocalizedText displayName;
    UA_LocalizedText description;
    UA_UInt32 writeMask;
    UA_UInt32 userWriteMask;
    UA_Boolean containsNoLoops;
    UA_Byte eventNotifier;
} UA_ViewAttributes;

#define UA_TYPES_VIEWATTRIBUTES 29
```

5.5.31 RequestHeader

The header passed with every server request.

```
typedef struct {
    UA_NodeId authenticationToken;
    UA_DateTime timestamp;
    UA_UInt32 requestHandle;
    UA_UInt32 returnDiagnostics;
    UA_String auditEntryId;
    UA_UInt32 timeoutHint;
    UA_ExtensionObject additionalHeader;
} UA_RequestHeader;

#define UA_TYPES_REQUESTHEADER 30
```

5.5.32 MonitoredItemModifyResult

```
typedef struct {
    UA_StatusCode statusCode;
    UA_Double revisedSamplingInterval;
    UA_UInt32 revisedQueueSize;
    UA_ExtensionObject filterResult;
} UA_MonitoredItemModifyResult;

#define UA_TYPES_MONITOREDITEMMODIFYRESULT 31
```

5.5.33 ElementOperand

```
typedef struct {
    UA_UInt32 index;
} UA_ElementOperand;

#define UA_TYPES_ELEMENTOPERAND 32
```

5.5.34 CloseSecureChannelRequest

Closes a secure channel.

```
typedef struct {
    UA_RequestHeader requestHeader;
} UA_CloseSecureChannelRequest;

#define UA_TYPES_CLOSESECURECHANNELREQUEST 33
```

5.5.35 AddNodesResult

A result of an add node operation.

```
typedef struct {
    UA_StatusCode statusCode;
    UA_NodeId addedNodeId;
} UA_AddNodesResult;

#define UA_TYPES_ADDNODESRESULT 34
```

5.5.36 VariableAttributes

The attributes for a variable node.

```
typedef struct {
    UA_UInt32 specifiedAttributes;
    UA_LocalizedText displayName;
    UA_LocalizedText description;
    UA_UInt32 writeMask;
    UA_UInt32 userWriteMask;
    UA_Variant value;
    UA_NodeId dataType;
    UA_Int32 valueRank;
    size_t arrayDimensionsSize;
    UA_UInt32 *arrayDimensions;
    UA_Byte accessLevel;
    UA_Byte userAccessLevel;
```

(continues on next page)

```
    UA_Double minimumSamplingInterval;
    UA_Boolean historizing;
} UA_VariableAttributes;

#define UA_TYPES_VARIABLEATTRIBUTES 35
```

5.5.37 NotificationMessage

```
typedef struct {
    UA_UInt32 sequenceNumber;
    UA_DateTime publishTime;
    size_t notificationDataSize;
    UA_ExtensionObject *notificationData;
} UA_NotificationMessage;

#define UA_TYPES_NOTIFICATIONMESSAGE 36
```

5.5.38 FindServersOnNetworkRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_UInt32 startingRecordId;
    UA_UInt32 maxRecordsToReturn;
    size_t serverCapabilityFilterSize;
    UA_String *serverCapabilityFilter;
} UA_FindServersOnNetworkRequest;

#define UA_TYPES_FINDSERVERSONNETWORKREQUEST 37
```

5.5.39 EventFieldList

```
typedef struct {
    UA_UInt32 clientHandle;
    size_t eventFieldsSize;
    UA_Variant *eventFields;
} UA_EventFieldList;

#define UA_TYPES_EVENTFIELDLIST 38
```

5.5.40 MonitoringMode

```
typedef enum {
    UA_MONITORINGMODE_DISABLED = 0,
    UA_MONITORINGMODE_SAMPLING = 1,
    UA_MONITORINGMODE_REPORTING = 2,
    __UA_MONITORINGMODE_FORCE32BIT = 0x7fffffff
} UA_MonitoringMode;
UA_STATIC_ASSERT(sizeof(UA_MonitoringMode) == sizeof(UA_Int32), enum_must_be_
→32bit);

#define UA_TYPES_MONITORINGMODE 39
```

5.5.41 MdnsDiscoveryConfiguration

The discovery information needed for mDNS registration.

```
typedef struct {
    UA_String mdnsServerName;
    size_t serverCapabilitiesSize;
    UA_String *serverCapabilities;
} UA_MdnsDiscoveryConfiguration;

#define UA_TYPES_MDNSDISCOVERYCONFIGURATION 40
```

5.5.42 CallMethodResult

```
typedef struct {
    UA_StatusCode statusCode;
    size_t inputArgumentResultsSize;
    UA_StatusCode *inputArgumentResults;
    size_t inputArgumentDiagnosticInfosSize;
    UA_DiagnosticInfo *inputArgumentDiagnosticInfos;
    size_t outputArgumentsSize;
    UA_Variant *outputArguments;
} UA_CallMethodResult;

#define UA_TYPES_CALLMETHODRESULT 41
```

5.5.43 ParsingResult

```
typedef struct {
    UA_StatusCode statusCode;
    size_t dataStatusCodesSize;
    UA_StatusCode *dataStatusCodes;
    size_t dataDiagnosticInfosSize;
    UA_DiagnosticInfo *dataDiagnosticInfos;
} UA_ParsingResult;

#define UA_TYPES_PARSINGRESULT 42
```

5.5.44 RelativePathElement

An element in a relative path.

```
typedef struct {
    UA_NodeId referenceTypeId;
    UA_Boolean isInverse;
    UA_Boolean includeSubtypes;
    UA_QualifiedName targetName;
} UA_RelativePathElement;

#define UA_TYPES_RELATIVEPATHELEMENT 43
```

5.5.45 BrowseDirection

The directions of the references to return.

```
typedef enum {
    UA_BROWSEDIRECTION_FORWARD = 0,
    UA_BROWSEDIRECTION_INVERSE = 1,
    UA_BROWSEDIRECTION_BOTH = 2,
    UA_BROWSEDIRECTION_INVALID = 3,
    __UA_BROWSEDIRECTION_FORCE32BIT = 0x7fffffff
} UA_BrowseDirection;
UA_STATIC_ASSERT(sizeof(UA_BrowseDirection) == sizeof(UA_Int32), enum_must_be_
→32bit);

#define UA_TYPES_BROWSEDIRECTION 44
```

5.5.46 CallMethodRequest

```
typedef struct {
    UA_NodeId objectId;
    UA_NodeId methodId;
    size_t inputArgumentsSize;
    UA_Variant *inputArguments;
} UA_CallMethodRequest;

#define UA_TYPES_CALLMETHODREQUEST 45
```

5.5.47 RedundancySupport

```
typedef enum {
    UA_REDUNDANCYSUPPORT_NONE = 0,
    UA_REDUNDANCYSUPPORT_COLD = 1,
    UA_REDUNDANCYSUPPORT_WARM = 2,
    UA_REDUNDANCYSUPPORT_HOT = 3,
    UA_REDUNDANCYSUPPORT_TRANSPARENT = 4,
    UA_REDUNDANCYSUPPORT_HOTANDMIRRORED = 5,
    __UA_REDUNDANCYSUPPORT_FORCE32BIT = 0x7fffffff
} UA_RedundancySupport;
UA_STATIC_ASSERT(sizeof(UA_RedundancySupport) == sizeof(UA_Int32), enum_must_be_
→32bit);

#define UA_TYPES_REDUNDANCYSUPPORT 46
```

5.5.48 EventNotificationList

```
typedef struct {
    size_t eventsSize;
    UA_EventFieldList *events;
} UA_EventNotificationList;

#define UA_TYPES_EVENTNOTIFICATIONLIST 47
```

5.5.49 UnregisterNodesRequest

Unregisters one or more previously registered nodes.

```
typedef struct {
    UA_RequestHeader requestHeader;
    size_t nodesToUnregisterSize;
    UA_NodeId *nodesToUnregister;
} UA_UnregisterNodesRequest;

#define UA_TYPES_UNREGISTERNODESREQUEST 48
```

5.5.50 ContentFilterElementResult

```
typedef struct {
    UA_StatusCode statusCode;
    size_t operandStatusCodesSize;
    UA_StatusCode *operandStatusCodes;
    size_t operandDiagnosticInfosSize;
    UA_DiagnosticInfo *operandDiagnosticInfos;
} UA_ContentFilterElementResult;

#define UA_TYPES_CONTENTFILTERELEMENTRESULT 49
```

5.5.51 SimpleAttributeOperand

```
typedef struct {
    UA_NodeId typeDefinitionId;
    size_t browsePathSize;
    UA_QualifiedName *browsePath;
    UA_UInt32 attributeId;
    UA_String indexRange;
} UA_SimpleAttributeOperand;

#define UA_TYPES_SIMPLEATTRIBUTEOPERAND 50
```

5.5.52 LiteralOperand

```
typedef struct {
    UA_Variant value;
} UA_LiteralOperand;

#define UA_TYPES_LITERALOPERAND 51
```

5.5.53 QueryDataSet

```
typedef struct {
    UA_ExpandedNodeId nodeId;
    UA_ExpandedNodeId typeDefinitionNode;
    size_t valuesSize;
    UA_Variant *values;
} UA_QueryDataSet;

#define UA_TYPES_QUERYDATASET 52
```

5.5.54 AnonymousIdentityToken

A token representing an anonymous user.

```
typedef struct {
    UA_String policyId;
} UA_AnonymousIdentityToken;

#define UA_TYPES_ANONYMOUSIDENTITYTOKEN 53
```

5.5.55 SetPublishingModeRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_Boolean publishingEnabled;
    size_t subscriptionIdsSize;
    UA_UInt32 *subscriptionIds;
} UA_SetPublishingModeRequest;

#define UA_TYPES_SETPUBLISHINGMODEREQUEST 54
```

5.5.56 MonitoredItemCreateResult

```
typedef struct {
    UA_StatusCode statusCode;
    UA_UInt32 monitoredItemId;
    UA_Double revisedSamplingInterval;
    UA_UInt32 revisedQueueSize;
    UA_ExtensionObject filterResult;
} UA_MonitoredItemCreateResult;

#define UA_TYPES_MONITOREDITEMCREATERESULT 55
```

5.5.57 TimestampsToReturn

```
typedef enum {
    UA_TIMESTAMPSTORETURN_SOURCE = 0,
    UA_TIMESTAMPSTORETURN_SERVER = 1,
    UA_TIMESTAMPSTORETURN_BOTH = 2,
    UA_TIMESTAMPSTORETURN_NEITHER = 3,
    UA_TIMESTAMPSTORETURN_INVALID = 4,
    __UA_TIMESTAMPSTORETURN_FORCE32BIT = 0x7fffffff
} UA_TimestampsToReturn;
UA_STATIC_ASSERT(sizeof(UA_TimestampsToReturn) == sizeof(UA_Int32), enum_must_be_
→32bit);

#define UA_TYPES_TIMESTAMPSTORETURN 56
```

5.5.58 CallRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    size_t methodsToCallSize;
    UA_CallMethodRequest *methodsToCall;
```

(continues on next page)

```
} UA_CallRequest;

#define UA_TYPES_CALLREQUEST 57
```

5.5.59 MethodAttributes

The attributes for a method node.

```
typedef struct {
    UA_UInt32 specifiedAttributes;
    UA_LocalizedText displayName;
    UA_LocalizedText description;
    UA_UInt32 writeMask;
    UA_UInt32 userWriteMask;
    UA_Boolean executable;
    UA_Boolean userExecutable;
} UA_MethodAttributes;

#define UA_TYPES_METHODATTRIBUTES 58
```

5.5.60 DeleteReferencesItem

A request to delete a node from the server address space.

```
typedef struct {
    UA_NodeId sourceNodeId;
    UA_NodeId referenceTypeId;
    UA_Boolean isForward;
    UA_ExpandedNodeId targetNodeId;
    UA_Boolean deleteBidirectional;
} UA_DeleteReferencesItem;

#define UA_TYPES_DELETEREFERENCESITEM 59
```

5.5.61 WriteValue

```
typedef struct {
    UA_NodeId nodeId;
    UA_UInt32 attributeId;
    UA_String indexRange;
    UA_DataValue value;
} UA_WriteValue;

#define UA_TYPES_WRITEVALUE 60
```

5.5.62 NodeAttributesMask

The bits used to specify default attributes for a new node.

```
typedef enum {
    UA_NODEATTRIBUTESMASK_NONE = 0,
    UA_NODEATTRIBUTESMASK_ACCESSLEVEL = 1,
    UA_NODEATTRIBUTESMASK_ARRAYDIMENSIONS = 2,
    UA_NODEATTRIBUTESMASK_BROWSENAME = 4,
```

```
    UA_NODEATTRIBUTESMASK_CONTAINSNOLOOPS = 8,
    UA_NODEATTRIBUTESMASK_DATATYPE = 16,
    UA_NODEATTRIBUTESMASK_DESCRIPTION = 32,
    UA_NODEATTRIBUTESMASK_DISPLAYNAME = 64,
    UA_NODEATTRIBUTESMASK_EVENTNOTIFIER = 128,
    UA_NODEATTRIBUTESMASK_EXECUTABLE = 256,
    UA_NODEATTRIBUTESMASK_HISTORIZING = 512,
    UA_NODEATTRIBUTESMASK_INVERSENAME = 1024,
    UA_NODEATTRIBUTESMASK_ISABSTRACT = 2048,
    UA_NODEATTRIBUTESMASK_MINIMUMSAMPLINGINTERVAL = 4096,
    UA_NODEATTRIBUTESMASK_NODECLASS = 8192,
    UA_NODEATTRIBUTESMASK_NODEID = 16384,
    UA_NODEATTRIBUTESMASK_SYMMETRIC = 32768,
    UA_NODEATTRIBUTESMASK_USERACCESSLEVEL = 65536,
    UA_NODEATTRIBUTESMASK_USEREXECUTABLE = 131072,
    UA_NODEATTRIBUTESMASK_USERWRITEMASK = 262144,
    UA_NODEATTRIBUTESMASK_VALUERANK = 524288,
    UA_NODEATTRIBUTESMASK_WRITEMASK = 1048576,
    UA_NODEATTRIBUTESMASK_VALUE = 2097152,
    UA_NODEATTRIBUTESMASK_DATATYPEDEFINITION = 4194304,
    UA_NODEATTRIBUTESMASK_ROLEPERMISSIONS = 8388608,
    UA_NODEATTRIBUTESMASK_ACCESSRESTRICTIONS = 16777216,
    UA_NODEATTRIBUTESMASK_ALL = 33554431,
    UA_NODEATTRIBUTESMASK_BASENODE = 26501220,
    UA_NODEATTRIBUTESMASK_OBJECT = 26501348,
    UA_NODEATTRIBUTESMASK_OBJECTTYPE = 26503268,
    UA_NODEATTRIBUTESMASK_VARIABLE = 26571383,
    UA_NODEATTRIBUTESMASK_VARIABLETYPE = 28600438,
    UA_NODEATTRIBUTESMASK_METHOD = 26632548,
    UA_NODEATTRIBUTESMASK_REFERENCETYPE = 26537060,
    UA_NODEATTRIBUTESMASK_VIEW = 26501356,
    __UA_NODEATTRIBUTESMASK_FORCE32BIT = 0x7fffffff
} UA_NodeAttributesMask;
UA_STATIC_ASSERT(sizeof(UA_NodeAttributesMask) == sizeof(UA_Int32), enum_must_be_
↪32bit);

#define UA_TYPES_NODEATTRIBUTESMASK 61
```

5.5.63 MessageSecurityMode

The type of security to use on a message.

```
typedef enum {
    UA_MESSAGESECURITYMODE_INVALID = 0,
    UA_MESSAGESECURITYMODE_NONE = 1,
    UA_MESSAGESECURITYMODE_SIGN = 2,
    UA_MESSAGESECURITYMODE_SIGNANDENCRYPT = 3,
    __UA_MESSAGESECURITYMODE_FORCE32BIT = 0x7fffffff
} UA_MessageSecurityMode;
UA_STATIC_ASSERT(sizeof(UA_MessageSecurityMode) == sizeof(UA_Int32), enum_must_be_
↪32bit);

#define UA_TYPES_MESSAGESECURITYMODE 62
```

5.5.64 MonitoringParameters

```
typedef struct {
    UA_UInt32 clientHandle;
    UA_Double samplingInterval;
    UA_ExtensionObject filter;
    UA_UInt32 queueSize;
    UA_Boolean discardOldest;
} UA_MonitoringParameters;

#define UA_TYPES_MONITORINGPARAMETERS 63
```

5.5.65 ReferenceNode

Specifies a reference which belongs to a node.

```
typedef struct {
    UA_NodeId referenceTypeId;
    UA_Boolean isInverse;
    UA_ExpandedNodeId targetId;
} UA_ReferenceNode;

#define UA_TYPES_REFERENCENODE 64
```

5.5.66 Argument

An argument for a method.

```
typedef struct {
    UA_String name;
    UA_NodeId dataType;
    UA_Int32 valueRank;
    size_t arrayDimensionsSize;
    UA_UInt32 *arrayDimensions;
    UA_LocalizedText description;
} UA_Argument;

#define UA_TYPES_ARGUMENT 65
```

5.5.67 ChannelSecurityToken

The token that identifies a set of keys for an active secure channel.

```
typedef struct {
    UA_UInt32 channelId;
    UA_UInt32 tokenId;
    UA_DateTime createdAt;
    UA_UInt32 revisedLifetime;
} UA_ChannelSecurityToken;

#define UA_TYPES_CHANNELSECURITYTOKEN 66
```

5.5.68 UserIdentityToken

A base type for a user identity token.

```
typedef struct {
    UA_String policyId;
} UA_UserIdentityToken;

#define UA_TYPES_USERIDENTITYTOKEN 67
```

5.5.69 SignatureData

A digital signature.

```
typedef struct {
    UA_String algorithm;
    UA_ByteString signature;
} UA_SignatureData;

#define UA_TYPES_SIGNATUREDATA 68
```

5.5.70 ObjectTypeAttributes

The attributes for an object type node.

```
typedef struct {
    UA_UInt32 specifiedAttributes;
    UA_LocalizedText displayName;
    UA_LocalizedText description;
    UA_UInt32 writeMask;
    UA_UInt32 userWriteMask;
    UA_Boolean isAbstract;
} UA_ObjectTypeAttributes;

#define UA_TYPES_OBJECTTYPEATTRIBUTES 69
```

5.5.71 DeadbandType

```
typedef enum {
    UA_DEADBANDTYPE_NONE = 0,
    UA_DEADBANDTYPE_ABSOLUTE = 1,
    UA_DEADBANDTYPE_PERCENT = 2,
    __UA_DEADBANDTYPE_FORCE32BIT = 0x7fffffff
} UA_DeadbandType;
UA_STATIC_ASSERT(sizeof(UA_DeadbandType) == sizeof(UA_Int32), enum_must_be_32bit);

#define UA_TYPES_DEADBANDTYPE 70
```

5.5.72 SecurityTokenRequestType

Indicates whether a token if being created or renewed.

```
typedef enum {
    UA_SECURITYTOKENREQUESTTYPE_ISSUE = 0,
    UA_SECURITYTOKENREQUESTTYPE_RENEW = 1,
    __UA_SECURITYTOKENREQUESTTYPE_FORCE32BIT = 0x7fffffff
} UA_SecurityTokenRequestType;
UA_STATIC_ASSERT(sizeof(UA_SecurityTokenRequestType) == sizeof(UA_Int32), enum_
↪must_be_32bit);
```

```
#define UA_TYPES_SECURITYTOKENREQUESTTYPE 71
```

5.5.73 NodeAttributes

The base attributes for all nodes.

```
typedef struct {
    UA_UInt32 specifiedAttributes;
    UA_LocalizedText displayName;
    UA_LocalizedText description;
    UA_UInt32 writeMask;
    UA_UInt32 userWriteMask;
} UA_NodeAttributes;

#define UA_TYPES_NODEATTRIBUTES 72
```

5.5.74 DataChangeTrigger

```
typedef enum {
    UA_DATACHANGETRIGGER_STATUS = 0,
    UA_DATACHANGETRIGGER_STATUSVALUE = 1,
    UA_DATACHANGETRIGGER_STATUSVALUETIMESTAMP = 2,
    __UA_DATACHANGETRIGGER_FORCE32BIT = 0x7fffffff
} UA_DataChangeTrigger;
UA_STATIC_ASSERT(sizeof(UA_DataChangeTrigger) == sizeof(UA_Int32), enum_must_be_
↪32bit);

#define UA_TYPES_DATACHANGETRIGGER 73
```

5.5.75 BuildInfo

```
typedef struct {
    UA_String productUri;
    UA_String manufacturerName;
    UA_String productName;
    UA_String softwareVersion;
    UA_String buildNumber;
    UA_DateTime buildDate;
} UA_BuildInfo;

#define UA_TYPES_BUILDINFO 74
```

5.5.76 NodeClass

A mask specifying the class of the node.

```
typedef enum {
    UA_NODECLASS_UNSPECIFIED = 0,
    UA_NODECLASS_OBJECT = 1,
    UA_NODECLASS_VARIABLE = 2,
    UA_NODECLASS_METHOD = 4,
    UA_NODECLASS_OBJECTTYPE = 8,
    UA_NODECLASS_VARIABLETYPE = 16,
```

```
    UA_NODECLASS_REFERENCETYPE = 32,
    UA_NODECLASS_DATATYPE = 64,
    UA_NODECLASS_VIEW = 128,
    __UA_NODECLASS_FORCE32BIT = 0x7fffffff
} UA_NodeClass;
UA_STATIC_ASSERT(sizeof(UA_NodeClass) == sizeof(UA_Int32), enum_must_be_32bit);

#define UA_TYPES_NODECLASS 75
```

5.5.77 SubscriptionDiagnosticsDataType

```
typedef struct {
    UA_NodeId sessionId;
    UA_UInt32 subscriptionId;
    UA_Byte priority;
    UA_Double publishingInterval;
    UA_UInt32 maxKeepAliveCount;
    UA_UInt32 maxLifetimeCount;
    UA_UInt32 maxNotificationsPerPublish;
    UA_Boolean publishingEnabled;
    UA_UInt32 modifyCount;
    UA_UInt32 enableCount;
    UA_UInt32 disableCount;
    UA_UInt32 republishRequestCount;
    UA_UInt32 republishMessageRequestCount;
    UA_UInt32 republishMessageCount;
    UA_UInt32 transferRequestCount;
    UA_UInt32 transferredToAltClientCount;
    UA_UInt32 transferredToSameClientCount;
    UA_UInt32 publishRequestCount;
    UA_UInt32 dataChangeNotificationsCount;
    UA_UInt32 eventNotificationsCount;
    UA_UInt32 notificationsCount;
    UA_UInt32 latePublishRequestCount;
    UA_UInt32 currentKeepAliveCount;
    UA_UInt32 currentLifetimeCount;
    UA_UInt32 unacknowledgedMessageCount;
    UA_UInt32 discardedMessageCount;
    UA_UInt32 monitoredItemCount;
    UA_UInt32 disabledMonitoredItemCount;
    UA_UInt32 monitoringQueueOverflowCount;
    UA_UInt32 nextSequenceNumber;
    UA_UInt32 eventQueueOverFlowCount;
} UA_SubscriptionDiagnosticsDataType;

#define UA_TYPES_SUBSCRIPTIONDIAGNOSTICSDATATYPE 76
```

5.5.78 FilterOperand

```
typedef void * UA_FilterOperand;

#define UA_TYPES_FILTEROPERAND 77
```

5.5.79 MonitoredItemNotification

```
typedef struct {
    UA_UInt32 clientHandle;
    UA_DataValue value;
} UA_MonitoredItemNotification;

#define UA_TYPES_MONITOREDITEMNOTIFICATION 78
```

5.5.80 DeleteNodesItem

A request to delete a node to the server address space.

```
typedef struct {
    UA_NodeId nodeId;
    UA_Boolean deleteTargetReferences;
} UA_DeleteNodesItem;

#define UA_TYPES_DELETENODESITEM 79
```

5.5.81 DeleteSubscriptionsRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    size_t subscriptionIdsSize;
    UA_UInt32 *subscriptionIds;
} UA_DeleteSubscriptionsRequest;

#define UA_TYPES_DELETESUBSCRIPTIONSREQUEST 80
```

5.5.82 SubscriptionAcknowledgement

```
typedef struct {
    UA_UInt32 subscriptionId;
    UA_UInt32 sequenceNumber;
} UA_SubscriptionAcknowledgement;

#define UA_TYPES_SUBSCRIPTIONACKNOWLEDGEMENT 81
```

5.5.83 ReadValueId

```
typedef struct {
    UA_NodeId nodeId;
    UA_UInt32 attributeId;
    UA_String indexRange;
    UA_QualifiedName dataEncoding;
} UA_ReadValueId;

#define UA_TYPES_READVALUEID 82
```

5.5.84 DataTypeAttributes

The attributes for a data type node.

```
typedef struct {
    UA_UInt32 specifiedAttributes;
    UA_LocalizedText displayName;
    UA_LocalizedText description;
    UA_UInt32 writeMask;
    UA_UInt32 userWriteMask;
    UA_Boolean isAbstract;
} UA_DataTypeAttributes;

#define UA_TYPES_DATATYPEATTRIBUTES 83
```

5.5.85 ResponseHeader

The header passed with every server response.

```
typedef struct {
    UA_DateTime timestamp;
    UA_UInt32 requestHandle;
    UA_StatusCode serviceResult;
    UA_DiagnosticInfo serviceDiagnostics;
    size_t stringTableSize;
    UA_String *stringTable;
    UA_ExtensionObject additionalHeader;
} UA_ResponseHeader;

#define UA_TYPES_RESPONSEHEADER 84
```

5.5.86 DeleteMonitoredItemsRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_UInt32 subscriptionId;
    size_t monitoredItemIdsSize;
    UA_UInt32 *monitoredItemIds;
} UA_DeleteMonitoredItemsRequest;

#define UA_TYPES_DELETEMONITOREDITEMSREQUEST 85
```

5.5.87 ViewDescription

The view to browse.

```
typedef struct {
    UA_NodeId viewId;
    UA_DateTime timestamp;
    UA_UInt32 viewVersion;
} UA_ViewDescription;

#define UA_TYPES_VIEWDESCRIPTION 86
```

5.5.88 ServerOnNetwork

```
typedef struct {
    UA_UInt32 recordId;
```

```
    UA_String serverName;
    UA_String discoveryUrl;
    size_t serverCapabilitiesSize;
    UA_String *serverCapabilities;
} UA_ServerOnNetwork;

#define UA_TYPES_SERVERONNETWORK 87
```

5.5.89 DeleteMonitoredItemsResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_StatusCode *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_DeleteMonitoredItemsResponse;

#define UA_TYPES_DELETEMONITOREDITEMSRESPONSE 88
```

5.5.90 FindServersOnNetworkResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    UA_DateTime lastCounterResetTime;
    size_t serversSize;
    UA_ServerOnNetwork *servers;
} UA_FindServersOnNetworkResponse;

#define UA_TYPES_FINDSERVERSONNETWORKRESPONSE 89
```

5.5.91 RelativePath

A relative path constructed from reference types and browse names.

```
typedef struct {
    size_t elementsSize;
    UA_RelativePathElement *elements;
} UA_RelativePath;

#define UA_TYPES_RELATIVEPATH 90
```

5.5.92 RegisterNodesRequest

Registers one or more nodes for repeated use within a session.

```
typedef struct {
    UA_RequestHeader requestHeader;
    size_t nodesToRegisterSize;
    UA_NodeId *nodesToRegister;
} UA_RegisterNodesRequest;

#define UA_TYPES_REGISTERNODESREQUEST 91
```

5.5.93 AggregateConfiguration

```
typedef struct {
    UA_Boolean useServerCapabilitiesDefaults;
    UA_Boolean treatUncertainAsBad;
    UA_Byte percentDataBad;
    UA_Byte percentDataGood;
    UA_Boolean useSlopedExtrapolation;
} UA_AggregateConfiguration;

#define UA_TYPES_AGGREGATECONFIGURATION 92
```

5.5.94 DeleteNodesRequest

Delete one or more nodes from the server address space.

```
typedef struct {
    UA_RequestHeader requestHeader;
    size_t nodesToDeleteSize;
    UA_DeleteNodesItem *nodesToDelete;
} UA_DeleteNodesRequest;

#define UA_TYPES_DELETENODESREQUEST 93
```

5.5.95 PublishResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    UA_UInt32 subscriptionId;
    size_t availableSequenceNumbersSize;
    UA_UInt32 *availableSequenceNumbers;
    UA_Boolean moreNotifications;
    UA_NotificationMessage notificationMessage;
    size_t resultsSize;
    UA_StatusCode *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_PublishResponse;

#define UA_TYPES_PUBLISHRESPONSE 94
```

5.5.96 MonitoredItemModifyRequest

```
typedef struct {
    UA_UInt32 monitoredItemId;
    UA_MonitoringParameters requestedParameters;
} UA_MonitoredItemModifyRequest;

#define UA_TYPES_MONITOREDITEMMODIFYREQUEST 95
```

5.5.97 ServiceCounterDataType

```
typedef struct {
    UA_UInt32 totalCount;
    UA_UInt32 errorCount;
} UA_ServiceCounterDataType;

#define UA_TYPES_SERVICECOUNTERDATATYPE 96
```

5.5.98 ModelChangeStructureDataType

```
typedef struct {
    UA_NodeId affected;
    UA_NodeId affectedType;
    UA_Byte verb;
} UA_ModelChangeStructureDataType;

#define UA_TYPES_MODELCHANGESTRUCTUREDATATYPE 97
```

5.5.99 UserNameIdentityToken

A token representing a user identified by a user name and password.

```
typedef struct {
    UA_String policyId;
    UA_String userName;
    UA_ByteString password;
    UA_String encryptionAlgorithm;
} UA_UserNameIdentityToken;

#define UA_TYPES_USERNAMEIDENTITYTOKEN 98
```

5.5.100 IdType

The type of identifier used in a node id.

```
typedef enum {
    UA_IDTYPE_NUMERIC = 0,
    UA_IDTYPE_STRING = 1,
    UA_IDTYPE_GUID = 2,
    UA_IDTYPE_OPAQUE = 3,
    __UA_IDTYPE_FORCE32BIT = 0x7fffffff
} UA_IdType;
UA_STATIC_ASSERT(sizeof(UA_IdType) == sizeof(UA_Int32), enum_must_be_32bit);

#define UA_TYPES_IDTYPE 99
```

5.5.101 UserTokenType

The possible user token types.

```
typedef enum {
    UA_USERTOKENTYPE_ANONYMOUS = 0,
    UA_USERTOKENTYPE_USERNAME = 1,
    UA_USERTOKENTYPE_CERTIFICATE = 2,
    UA_USERTOKENTYPE_ISSUEDTOKEN = 3,
    __UA_USERTOKENTYPE_FORCE32BIT = 0x7fffffff
```

(continues on next page)

```
} UA_UserTokenType;
UA_STATIC_ASSERT(sizeof(UA_UserTokenType) == sizeof(UA_Int32), enum_must_be_32bit);

#define UA_TYPES_USERTOKENTYPE 100
```

5.5.102 SetTriggeringResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t addResultsSize;
    UA_StatusCode *addResults;
    size_t addDiagnosticInfosSize;
    UA_DiagnosticInfo *addDiagnosticInfos;
    size_t removeResultsSize;
    UA_StatusCode *removeResults;
    size_t removeDiagnosticInfosSize;
    UA_DiagnosticInfo *removeDiagnosticInfos;
} UA_SetTriggeringResponse;

#define UA_TYPES_SETTRIGGERINGRESPONSE 101
```

5.5.103 TimeZoneDataType

```
typedef struct {
    UA_Int16 offset;
    UA_Boolean daylightSavingInOffset;
} UA_TimeZoneDataType;

#define UA_TYPES_TIMEZONEDATATYPE 102
```

5.5.104 ActivateSessionRequest

Activates a session with the server.

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_SignatureData clientSignature;
    size_t clientSoftwareCertificatesSize;
    UA_SignedSoftwareCertificate *clientSoftwareCertificates;
    size_t localeIdsSize;
    UA_String *localeIds;
    UA_ExtensionObject userIdentityToken;
    UA_SignatureData userTokenSignature;
} UA_ActivateSessionRequest;

#define UA_TYPES_ACTIVATESESSIONREQUEST 103
```

5.5.105 OpenSecureChannelResponse

Creates a secure channel with a server.

```
typedef struct {
    UA_ResponseHeader responseHeader;
    UA_UInt32 serverProtocolVersion;
```

```
    UA_ChannelSecurityToken securityToken;
    UA_ByteString serverNonce;
} UA_OpenSecureChannelResponse;

#define UA_TYPES_OPENSECURECHANNELRESPONSE 104
```

5.5.106 ApplicationType

The types of applications.

```
typedef enum {
    UA_APPLICATIONTYPE_SERVER = 0,
    UA_APPLICATIONTYPE_CLIENT = 1,
    UA_APPLICATIONTYPE_CLIENTANDSERVER = 2,
    UA_APPLICATIONTYPE_DISCOVERYSERVER = 3,
    __UA_APPLICATIONTYPE_FORCE32BIT = 0x7fffffff
} UA_ApplicationType;
UA_STATIC_ASSERT(sizeof(UA_ApplicationType) == sizeof(UA_Int32), enum_must_be_
→32bit);

#define UA_TYPES_APPLICATIONTYPE 105
```

5.5.107 ServerState

```
typedef enum {
    UA_SERVERSTATE_RUNNING = 0,
    UA_SERVERSTATE_FAILED = 1,
    UA_SERVERSTATE_NOCONFIGURATION = 2,
    UA_SERVERSTATE_SUSPENDED = 3,
    UA_SERVERSTATE_SHUTDOWN = 4,
    UA_SERVERSTATE_TEST = 5,
    UA_SERVERSTATE_COMMUNICATIONFAULT = 6,
    UA_SERVERSTATE_UNKNOWN = 7,
    __UA_SERVERSTATE_FORCE32BIT = 0x7fffffff
} UA_ServerState;
UA_STATIC_ASSERT(sizeof(UA_ServerState) == sizeof(UA_Int32), enum_must_be_32bit);

#define UA_TYPES_SERVERSTATE 106
```

5.5.108 QueryNextResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t queryDataSetsSize;
    UA_QueryDataSet *queryDataSets;
    UA_ByteString revisedContinuationPoint;
} UA_QueryNextResponse;

#define UA_TYPES_QUERYNEXTRESPONSE 107
```

5.5.109 DiscoveryConfiguration

A base type for discovery configuration information.

```c
typedef void * UA_DiscoveryConfiguration;

#define UA_TYPES_DISCOVERYCONFIGURATION 108
```

5.5.110 ActivateSessionResponse

Activates a session with the server.

```c
typedef struct {
    UA_ResponseHeader responseHeader;
    UA_ByteString serverNonce;
    size_t resultsSize;
    UA_StatusCode *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_ActivateSessionResponse;

#define UA_TYPES_ACTIVATESESSIONRESPONSE 109
```

5.5.111 EndpointUrlListDataType

```c
typedef struct {
    size_t endpointUrlListSize;
    UA_String *endpointUrlList;
} UA_EndpointUrlListDataType;

#define UA_TYPES_ENDPOINTURLLISTDATATYPE 110
```

5.5.112 FilterOperator

```c
typedef enum {
    UA_FILTEROPERATOR_EQUALS = 0,
    UA_FILTEROPERATOR_ISNULL = 1,
    UA_FILTEROPERATOR_GREATERTHAN = 2,
    UA_FILTEROPERATOR_LESSTHAN = 3,
    UA_FILTEROPERATOR_GREATERTHANOREQUAL = 4,
    UA_FILTEROPERATOR_LESSTHANOREQUAL = 5,
    UA_FILTEROPERATOR_LIKE = 6,
    UA_FILTEROPERATOR_NOT = 7,
    UA_FILTEROPERATOR_BETWEEN = 8,
    UA_FILTEROPERATOR_INLIST = 9,
    UA_FILTEROPERATOR_AND = 10,
    UA_FILTEROPERATOR_OR = 11,
    UA_FILTEROPERATOR_CAST = 12,
    UA_FILTEROPERATOR_INVIEW = 13,
    UA_FILTEROPERATOR_OFTYPE = 14,
    UA_FILTEROPERATOR_RELATEDTO = 15,
    UA_FILTEROPERATOR_BITWISEAND = 16,
    UA_FILTEROPERATOR_BITWISEOR = 17,
    __UA_FILTEROPERATOR_FORCE32BIT = 0x7fffffff
} UA_FilterOperator;
UA_STATIC_ASSERT(sizeof(UA_FilterOperator) == sizeof(UA_Int32), enum_must_be_
↪32bit);

#define UA_TYPES_FILTEROPERATOR 111
```

5.5.113 QueryNextRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_Boolean releaseContinuationPoint;
    UA_ByteString continuationPoint;
} UA_QueryNextRequest;

#define UA_TYPES_QUERYNEXTREQUEST 112
```

5.5.114 WriteResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_StatusCode *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_WriteResponse;

#define UA_TYPES_WRITERESPONSE 113
```

5.5.115 BrowseNextRequest

Continues one or more browse operations.

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_Boolean releaseContinuationPoints;
    size_t continuationPointsSize;
    UA_ByteString *continuationPoints;
} UA_BrowseNextRequest;

#define UA_TYPES_BROWSENEXTREQUEST 114
```

5.5.116 CreateSubscriptionRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_Double requestedPublishingInterval;
    UA_UInt32 requestedLifetimeCount;
    UA_UInt32 requestedMaxKeepAliveCount;
    UA_UInt32 maxNotificationsPerPublish;
    UA_Boolean publishingEnabled;
    UA_Byte priority;
} UA_CreateSubscriptionRequest;

#define UA_TYPES_CREATESUBSCRIPTIONREQUEST 115
```

5.5.117 VariableTypeAttributes

The attributes for a variable type node.

```
typedef struct {
    UA_UInt32 specifiedAttributes;
    UA_LocalizedText displayName;
    UA_LocalizedText description;
    UA_UInt32 writeMask;
    UA_UInt32 userWriteMask;
    UA_Variant value;
    UA_NodeId dataType;
    UA_Int32 valueRank;
    size_t arrayDimensionsSize;
    UA_UInt32 *arrayDimensions;
    UA_Boolean isAbstract;
} UA_VariableTypeAttributes;

#define UA_TYPES_VARIABLETYPEATTRIBUTES 116
```

5.5.118 BrowsePathResult

The result of a translate opearation.

```
typedef struct {
    UA_StatusCode statusCode;
    size_t targetsSize;
    UA_BrowsePathTarget *targets;
} UA_BrowsePathResult;

#define UA_TYPES_BROWSEPATHRESULT 117
```

5.5.119 ModifySubscriptionResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    UA_Double revisedPublishingInterval;
    UA_UInt32 revisedLifetimeCount;
    UA_UInt32 revisedMaxKeepAliveCount;
} UA_ModifySubscriptionResponse;

#define UA_TYPES_MODIFYSUBSCRIPTIONRESPONSE 118
```

5.5.120 RedundantServerDataType

```
typedef struct {
    UA_String serverId;
    UA_Byte serviceLevel;
    UA_ServerState serverState;
} UA_RedundantServerDataType;

#define UA_TYPES_REDUNDANTSERVERDATATYPE 119
```

5.5.121 RegisterNodesResponse

Registers one or more nodes for repeated use within a session.

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t registeredNodeIdsSize;
    UA_NodeId *registeredNodeIds;
} UA_RegisterNodesResponse;

#define UA_TYPES_REGISTERNODESRESPONSE 120
```

5.5.122 CloseSessionRequest

Closes a session with the server.

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_Boolean deleteSubscriptions;
} UA_CloseSessionRequest;

#define UA_TYPES_CLOSESESSIONREQUEST 121
```

5.5.123 ModifyMonitoredItemsResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_MonitoredItemModifyResult *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_ModifyMonitoredItemsResponse;

#define UA_TYPES_MODIFYMONITOREDITEMSRESPONSE 122
```

5.5.124 ModifySubscriptionRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_UInt32 subscriptionId;
    UA_Double requestedPublishingInterval;
    UA_UInt32 requestedLifetimeCount;
    UA_UInt32 requestedMaxKeepAliveCount;
    UA_UInt32 maxNotificationsPerPublish;
    UA_Byte priority;
} UA_ModifySubscriptionRequest;

#define UA_TYPES_MODIFYSUBSCRIPTIONREQUEST 123
```

5.5.125 ServerDiagnosticsSummaryDataType

```
typedef struct {
    UA_UInt32 serverViewCount;
    UA_UInt32 currentSessionCount;
    UA_UInt32 cumulatedSessionCount;
    UA_UInt32 securityRejectedSessionCount;
    UA_UInt32 rejectedSessionCount;
    UA_UInt32 sessionTimeoutCount;
```

(continues on next page)

```
    UA_UInt32 sessionAbortCount;
    UA_UInt32 currentSubscriptionCount;
    UA_UInt32 cumulatedSubscriptionCount;
    UA_UInt32 publishingIntervalCount;
    UA_UInt32 securityRejectedRequestsCount;
    UA_UInt32 rejectedRequestsCount;
} UA_ServerDiagnosticsSummaryDataType;

#define UA_TYPES_SERVERDIAGNOSTICSSUMMARYDATATYPE 124
```

5.5.126 UserTokenPolicy

Describes a user token that can be used with a server.

```
typedef struct {
    UA_String policyId;
    UA_UserTokenType tokenType;
    UA_String issuedTokenType;
    UA_String issuerEndpointUrl;
    UA_String securityPolicyUri;
} UA_UserTokenPolicy;

#define UA_TYPES_USERTOKENPOLICY 125
```

5.5.127 ReferenceTypeAttributes

The attributes for a reference type node.

```
typedef struct {
    UA_UInt32 specifiedAttributes;
    UA_LocalizedText displayName;
    UA_LocalizedText description;
    UA_UInt32 writeMask;
    UA_UInt32 userWriteMask;
    UA_Boolean isAbstract;
    UA_Boolean symmetric;
    UA_LocalizedText inverseName;
} UA_ReferenceTypeAttributes;

#define UA_TYPES_REFERENCETYPEATTRIBUTES 126
```

5.5.128 BrowsePath

A request to translate a path into a node id.

```
typedef struct {
    UA_NodeId startingNode;
    UA_RelativePath relativePath;
} UA_BrowsePath;

#define UA_TYPES_BROWSEPATH 127
```

5.5.129 SetMonitoringModeRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_UInt32 subscriptionId;
    UA_MonitoringMode monitoringMode;
    size_t monitoredItemIdsSize;
    UA_UInt32 *monitoredItemIds;
} UA_SetMonitoringModeRequest;

#define UA_TYPES_SETMONITORINGMODEREQUEST 128
```

5.5.130 UnregisterNodesResponse

Unregisters one or more previously registered nodes.

```
typedef struct {
    UA_ResponseHeader responseHeader;
} UA_UnregisterNodesResponse;

#define UA_TYPES_UNREGISTERNODESRESPONSE 129
```

5.5.131 WriteRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    size_t nodesToWriteSize;
    UA_WriteValue *nodesToWrite;
} UA_WriteRequest;

#define UA_TYPES_WRITEREQUEST 130
```

5.5.132 ObjectAttributes

The attributes for an object node.

```
typedef struct {
    UA_UInt32 specifiedAttributes;
    UA_LocalizedText displayName;
    UA_LocalizedText description;
    UA_UInt32 writeMask;
    UA_UInt32 userWriteMask;
    UA_Byte eventNotifier;
} UA_ObjectAttributes;

#define UA_TYPES_OBJECTATTRIBUTES 131
```

5.5.133 BrowseResultMask

A bit mask which specifies what should be returned in a browse response.

```
typedef enum {
    UA_BROWSERESULTMASK_NONE = 0,
    UA_BROWSERESULTMASK_REFERENCETYPEID = 1,
    UA_BROWSERESULTMASK_ISFORWARD = 2,
```

(continues on next page)

```
    UA_BROWSERESULTMASK_NODECLASS = 4,
    UA_BROWSERESULTMASK_BROWSENAME = 8,
    UA_BROWSERESULTMASK_DISPLAYNAME = 16,
    UA_BROWSERESULTMASK_TYPEDEFINITION = 32,
    UA_BROWSERESULTMASK_ALL = 63,
    UA_BROWSERESULTMASK_REFERENCETYPEINFO = 3,
    UA_BROWSERESULTMASK_TARGETINFO = 60,
    __UA_BROWSERESULTMASK_FORCE32BIT = 0x7fffffff
} UA_BrowseResultMask;
UA_STATIC_ASSERT(sizeof(UA_BrowseResultMask) == sizeof(UA_Int32), enum_must_be_
↪32bit);

#define UA_TYPES_BROWSERESULTMASK 132
```

5.5.134 BrowseDescription

A request to browse the the references from a node.

```
typedef struct {
    UA_NodeId nodeId;
    UA_BrowseDirection browseDirection;
    UA_NodeId referenceTypeId;
    UA_Boolean includeSubtypes;
    UA_UInt32 nodeClassMask;
    UA_UInt32 resultMask;
} UA_BrowseDescription;

#define UA_TYPES_BROWSEDESCRIPTION 133
```

5.5.135 SetTriggeringRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_UInt32 subscriptionId;
    UA_UInt32 triggeringItemId;
    size_t linksToAddSize;
    UA_UInt32 *linksToAdd;
    size_t linksToRemoveSize;
    UA_UInt32 *linksToRemove;
} UA_SetTriggeringRequest;

#define UA_TYPES_SETTRIGGERINGREQUEST 134
```

5.5.136 SessionSecurityDiagnosticsDataType

```
typedef struct {
    UA_NodeId sessionId;
    UA_String clientUserIdOfSession;
    size_t clientUserIdHistorySize;
    UA_String *clientUserIdHistory;
    UA_String authenticationMechanism;
    UA_String encoding;
    UA_String transportProtocol;
    UA_MessageSecurityMode securityMode;
    UA_String securityPolicyUri;
```

```
    UA_ByteString clientCertificate;
} UA_SessionSecurityDiagnosticsDataType;

#define UA_TYPES_SESSIONSECURITYDIAGNOSTICSDATATYPE 135
```

5.5.137 RepublishRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_UInt32 subscriptionId;
    UA_UInt32 retransmitSequenceNumber;
} UA_RepublishRequest;

#define UA_TYPES_REPUBLISHREQUEST 136
```

5.5.138 GetEndpointsRequest

Gets the endpoints used by the server.

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_String endpointUrl;
    size_t localeIdsSize;
    UA_String *localeIds;
    size_t profileUrisSize;
    UA_String *profileUris;
} UA_GetEndpointsRequest;

#define UA_TYPES_GETENDPOINTSREQUEST 137
```

5.5.139 PublishRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    size_t subscriptionAcknowledgementsSize;
    UA_SubscriptionAcknowledgement *subscriptionAcknowledgements;
} UA_PublishRequest;

#define UA_TYPES_PUBLISHREQUEST 138
```

5.5.140 DeleteSubscriptionsResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_StatusCode *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_DeleteSubscriptionsResponse;

#define UA_TYPES_DELETESUBSCRIPTIONSRESPONSE 139
```

5.5.141 AddNodesResponse

Adds one or more nodes to the server address space.

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_AddNodesResult *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_AddNodesResponse;

#define UA_TYPES_ADDNODESRESPONSE 140
```

5.5.142 DataChangeNotification

```
typedef struct {
    size_t monitoredItemsSize;
    UA_MonitoredItemNotification *monitoredItems;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_DataChangeNotification;

#define UA_TYPES_DATACHANGENOTIFICATION 141
```

5.5.143 CloseSecureChannelResponse

Closes a secure channel.

```
typedef struct {
    UA_ResponseHeader responseHeader;
} UA_CloseSecureChannelResponse;

#define UA_TYPES_CLOSESECURECHANNELRESPONSE 142
```

5.5.144 ModifyMonitoredItemsRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_UInt32 subscriptionId;
    UA_TimestampsToReturn timestampsToReturn;
    size_t itemsToModifySize;
    UA_MonitoredItemModifyRequest *itemsToModify;
} UA_ModifyMonitoredItemsRequest;

#define UA_TYPES_MODIFYMONITOREDITEMSREQUEST 143
```

5.5.145 SetMonitoringModeResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_StatusCode *results;
    size_t diagnosticInfosSize;
```

```
    UA_DiagnosticInfo *diagnosticInfos;
} UA_SetMonitoringModeResponse;

#define UA_TYPES_SETMONITORINGMODERESPONSE 144
```

5.5.146 FindServersRequest

Finds the servers known to the discovery server.

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_String endpointUrl;
    size_t localeIdsSize;
    UA_String *localeIds;
    size_t serverUrisSize;
    UA_String *serverUris;
} UA_FindServersRequest;

#define UA_TYPES_FINDSERVERSREQUEST 145
```

5.5.147 ReferenceDescription

The description of a reference.

```
typedef struct {
    UA_NodeId referenceTypeId;
    UA_Boolean isForward;
    UA_ExpandedNodeId nodeId;
    UA_QualifiedName browseName;
    UA_LocalizedText displayName;
    UA_NodeClass nodeClass;
    UA_ExpandedNodeId typeDefinition;
} UA_ReferenceDescription;

#define UA_TYPES_REFERENCEDESCRIPTION 146
```

5.5.148 SetPublishingModeResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_StatusCode *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_SetPublishingModeResponse;

#define UA_TYPES_SETPUBLISHINGMODERESPONSE 147
```

5.5.149 ContentFilterResult

```
typedef struct {
    size_t elementResultsSize;
    UA_ContentFilterElementResult *elementResults;
    size_t elementDiagnosticInfosSize;
```

```
    UA_DiagnosticInfo *elementDiagnosticInfos;
} UA_ContentFilterResult;

#define UA_TYPES_CONTENTFILTERRESULT 148
```

5.5.150 RegisterServerResponse

Registers a server with the discovery server.

```
typedef struct {
    UA_ResponseHeader responseHeader;
} UA_RegisterServerResponse;

#define UA_TYPES_REGISTERSERVERRESPONSE 149
```

5.5.151 AddReferencesItem

A request to add a reference to the server address space.

```
typedef struct {
    UA_NodeId sourceNodeId;
    UA_NodeId referenceTypeId;
    UA_Boolean isForward;
    UA_String targetServerUri;
    UA_ExpandedNodeId targetNodeId;
    UA_NodeClass targetNodeClass;
} UA_AddReferencesItem;

#define UA_TYPES_ADDREFERENCESITEM 150
```

5.5.152 QueryDataDescription

```
typedef struct {
    UA_RelativePath relativePath;
    UA_UInt32 attributeId;
    UA_String indexRange;
} UA_QueryDataDescription;

#define UA_TYPES_QUERYDATADESCRIPTION 151
```

5.5.153 CreateSubscriptionResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    UA_UInt32 subscriptionId;
    UA_Double revisedPublishingInterval;
    UA_UInt32 revisedLifetimeCount;
    UA_UInt32 revisedMaxKeepAliveCount;
} UA_CreateSubscriptionResponse;

#define UA_TYPES_CREATESUBSCRIPTIONRESPONSE 152
```

5.5.154 NetworkGroupDataType

```
typedef struct {
    UA_String serverUri;
    size_t networkPathsSize;
    UA_EndpointUrlListDataType *networkPaths;
} UA_NetworkGroupDataType;

#define UA_TYPES_NETWORKGROUPDATATYPE 153
```

5.5.155 DeleteReferencesResponse

Delete one or more references from the server address space.

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_StatusCode *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_DeleteReferencesResponse;

#define UA_TYPES_DELETEREFERENCESRESPONSE 154
```

5.5.156 CreateMonitoredItemsResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_MonitoredItemCreateResult *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_CreateMonitoredItemsResponse;

#define UA_TYPES_CREATEMONITOREDITEMSRESPONSE 155
```

5.5.157 CallResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_CallMethodResult *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_CallResponse;

#define UA_TYPES_CALLRESPONSE 156
```

5.5.158 DeleteNodesResponse

Delete one or more nodes from the server address space.

```
typedef struct {
    UA_ResponseHeader responseHeader;
```

```
    size_t resultsSize;
    UA_StatusCode *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_DeleteNodesResponse;

#define UA_TYPES_DELETENODESRESPONSE 157
```

5.5.159 RepublishResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    UA_NotificationMessage notificationMessage;
} UA_RepublishResponse;

#define UA_TYPES_REPUBLISHRESPONSE 158
```

5.5.160 MonitoredItemCreateRequest

```
typedef struct {
    UA_ReadValueId itemToMonitor;
    UA_MonitoringMode monitoringMode;
    UA_MonitoringParameters requestedParameters;
} UA_MonitoredItemCreateRequest;

#define UA_TYPES_MONITOREDITEMCREATEREQUEST 159
```

5.5.161 DeleteReferencesRequest

Delete one or more references from the server address space.

```
typedef struct {
    UA_RequestHeader requestHeader;
    size_t referencesToDeleteSize;
    UA_DeleteReferencesItem *referencesToDelete;
} UA_DeleteReferencesRequest;

#define UA_TYPES_DELETEREFERENCESREQUEST 160
```

5.5.162 ReadResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_DataValue *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_ReadResponse;

#define UA_TYPES_READRESPONSE 161
```

5.5.163 AddReferencesRequest

Adds one or more references to the server address space.

```
typedef struct {
    UA_RequestHeader requestHeader;
    size_t referencesToAddSize;
    UA_AddReferencesItem *referencesToAdd;
} UA_AddReferencesRequest;

#define UA_TYPES_ADDREFERENCESREQUEST 162
```

5.5.164 ReadRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_Double maxAge;
    UA_TimestampsToReturn timestampsToReturn;
    size_t nodesToReadSize;
    UA_ReadValueId *nodesToRead;
} UA_ReadRequest;

#define UA_TYPES_READREQUEST 163
```

5.5.165 OpenSecureChannelRequest

Creates a secure channel with a server.

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_UInt32 clientProtocolVersion;
    UA_SecurityTokenRequestType requestType;
    UA_MessageSecurityMode securityMode;
    UA_ByteString clientNonce;
    UA_UInt32 requestedLifetime;
} UA_OpenSecureChannelRequest;

#define UA_TYPES_OPENSECURECHANNELREQUEST 164
```

5.5.166 RegisterServer2Response

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t configurationResultsSize;
    UA_StatusCode *configurationResults;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_RegisterServer2Response;

#define UA_TYPES_REGISTERSERVER2RESPONSE 165
```

5.5.167 AddNodesItem

A request to add a node to the server address space.

```
typedef struct {
    UA_ExpandedNodeId parentNodeId;
    UA_NodeId referenceTypeId;
    UA_ExpandedNodeId requestedNewNodeId;
    UA_QualifiedName browseName;
    UA_NodeClass nodeClass;
    UA_ExtensionObject nodeAttributes;
    UA_ExpandedNodeId typeDefinition;
} UA_AddNodesItem;

#define UA_TYPES_ADDNODESITEM 166
```

5.5.168 NodeTypeDescription

```
typedef struct {
    UA_ExpandedNodeId typeDefinitionNode;
    UA_Boolean includeSubTypes;
    size_t dataToReturnSize;
    UA_QueryDataDescription *dataToReturn;
} UA_NodeTypeDescription;

#define UA_TYPES_NODETYPEDESCRIPTION 167
```

5.5.169 ServerStatusDataType

```
typedef struct {
    UA_DateTime startTime;
    UA_DateTime currentTime;
    UA_ServerState state;
    UA_BuildInfo buildInfo;
    UA_UInt32 secondsTillShutdown;
    UA_LocalizedText shutdownReason;
} UA_ServerStatusDataType;

#define UA_TYPES_SERVERSTATUSDATATYPE 168
```

5.5.170 AttributeOperand

```
typedef struct {
    UA_NodeId nodeId;
    UA_String alias;
    UA_RelativePath browsePath;
    UA_UInt32 attributeId;
    UA_String indexRange;
} UA_AttributeOperand;

#define UA_TYPES_ATTRIBUTEOPERAND 169
```

5.5.171 AddReferencesResponse

Adds one or more references to the server address space.

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_StatusCode *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_AddReferencesResponse;

#define UA_TYPES_ADDREFERENCESRESPONSE 170
```

5.5.172 EventFilterResult

```
typedef struct {
    size_t selectClauseResultsSize;
    UA_StatusCode *selectClauseResults;
    size_t selectClauseDiagnosticInfosSize;
    UA_DiagnosticInfo *selectClauseDiagnosticInfos;
    UA_ContentFilterResult whereClauseResult;
} UA_EventFilterResult;

#define UA_TYPES_EVENTFILTERRESULT 171
```

5.5.173 TranslateBrowsePathsToNodeIdsResponse

Translates one or more paths in the server address space.

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_BrowsePathResult *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_TranslateBrowsePathsToNodeIdsResponse;

#define UA_TYPES_TRANSLATEBROWSEPATHSTONODEIDSRESPONSE 172
```

5.5.174 DataChangeFilter

```
typedef struct {
    UA_DataChangeTrigger trigger;
    UA_UInt32 deadbandType;
    UA_Double deadbandValue;
} UA_DataChangeFilter;

#define UA_TYPES_DATACHANGEFILTER 173
```

5.5.175 ContentFilterElement

```
typedef struct {
    UA_FilterOperator filterOperator;
    size_t filterOperandsSize;
    UA_ExtensionObject *filterOperands;
} UA_ContentFilterElement;

#define UA_TYPES_CONTENTFILTERELEMENT 174
```

5.5.176 TranslateBrowsePathsToNodeIdsRequest

Translates one or more paths in the server address space.

```
typedef struct {
    UA_RequestHeader requestHeader;
    size_t browsePathsSize;
    UA_BrowsePath *browsePaths;
} UA_TranslateBrowsePathsToNodeIdsRequest;

#define UA_TYPES_TRANSLATEBROWSEPATHSTONODEIDSREQUEST 175
```

5.5.177 CloseSessionResponse

Closes a session with the server.

```
typedef struct {
    UA_ResponseHeader responseHeader;
} UA_CloseSessionResponse;

#define UA_TYPES_CLOSESESSIONRESPONSE 176
```

5.5.178 ApplicationDescription

Describes an application and how to find it.

```
typedef struct {
    UA_String applicationUri;
    UA_String productUri;
    UA_LocalizedText applicationName;
    UA_ApplicationType applicationType;
    UA_String gatewayServerUri;
    UA_String discoveryProfileUri;
    size_t discoveryUrlsSize;
    UA_String *discoveryUrls;
} UA_ApplicationDescription;

#define UA_TYPES_APPLICATIONDESCRIPTION 177
```

5.5.179 SessionDiagnosticsDataType

```
typedef struct {
    UA_NodeId sessionId;
    UA_String sessionName;
    UA_ApplicationDescription clientDescription;
    UA_String serverUri;
    UA_String endpointUrl;
    size_t localeIdsSize;
    UA_String *localeIds;
    UA_Double actualSessionTimeout;
    UA_UInt32 maxResponseMessageSize;
    UA_DateTime clientConnectionTime;
    UA_DateTime clientLastContactTime;
    UA_UInt32 currentSubscriptionsCount;
    UA_UInt32 currentMonitoredItemsCount;
    UA_UInt32 currentPublishRequestsInQueue;
    UA_ServiceCounterDataType totalRequestCount;
```

(continues on next page)

```
    UA_UInt32 unauthorizedRequestCount;
    UA_ServiceCounterDataType readCount;
    UA_ServiceCounterDataType historyReadCount;
    UA_ServiceCounterDataType writeCount;
    UA_ServiceCounterDataType historyUpdateCount;
    UA_ServiceCounterDataType callCount;
    UA_ServiceCounterDataType createMonitoredItemsCount;
    UA_ServiceCounterDataType modifyMonitoredItemsCount;
    UA_ServiceCounterDataType setMonitoringModeCount;
    UA_ServiceCounterDataType setTriggeringCount;
    UA_ServiceCounterDataType deleteMonitoredItemsCount;
    UA_ServiceCounterDataType createSubscriptionCount;
    UA_ServiceCounterDataType modifySubscriptionCount;
    UA_ServiceCounterDataType setPublishingModeCount;
    UA_ServiceCounterDataType publishCount;
    UA_ServiceCounterDataType republishCount;
    UA_ServiceCounterDataType transferSubscriptionsCount;
    UA_ServiceCounterDataType deleteSubscriptionsCount;
    UA_ServiceCounterDataType addNodesCount;
    UA_ServiceCounterDataType addReferencesCount;
    UA_ServiceCounterDataType deleteNodesCount;
    UA_ServiceCounterDataType deleteReferencesCount;
    UA_ServiceCounterDataType browseCount;
    UA_ServiceCounterDataType browseNextCount;
    UA_ServiceCounterDataType translateBrowsePathsToNodeIdsCount;
    UA_ServiceCounterDataType queryFirstCount;
    UA_ServiceCounterDataType queryNextCount;
    UA_ServiceCounterDataType registerNodesCount;
    UA_ServiceCounterDataType unregisterNodesCount;
} UA_SessionDiagnosticsDataType;

#define UA_TYPES_SESSIONDIAGNOSTICSDATATYPE 178
```

5.5.180 ServiceFault

The response returned by all services when there is a service level error.

```
typedef struct {
    UA_ResponseHeader responseHeader;
} UA_ServiceFault;

#define UA_TYPES_SERVICEFAULT 179
```

5.5.181 RegisteredServer

The information required to register a server with a discovery server.

```
typedef struct {
    UA_String serverUri;
    UA_String productUri;
    size_t serverNamesSize;
    UA_LocalizedText *serverNames;
    UA_ApplicationType serverType;
    UA_String gatewayServerUri;
    size_t discoveryUrlsSize;
    UA_String *discoveryUrls;
    UA_String semaphoreFilePath;
    UA_Boolean isOnline;
```

```
} UA_RegisteredServer;

#define UA_TYPES_REGISTEREDSERVER 180
```

5.5.182 AggregateFilter

```
typedef struct {
    UA_DateTime startTime;
    UA_NodeId aggregateType;
    UA_Double processingInterval;
    UA_AggregateConfiguration aggregateConfiguration;
} UA_AggregateFilter;

#define UA_TYPES_AGGREGATEFILTER 181
```

5.5.183 RegisterServerRequest

Registers a server with the discovery server.

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_RegisteredServer server;
} UA_RegisterServerRequest;

#define UA_TYPES_REGISTERSERVERREQUEST 182
```

5.5.184 EndpointDescription

The description of a endpoint that can be used to access a server.

```
typedef struct {
    UA_String endpointUrl;
    UA_ApplicationDescription server;
    UA_ByteString serverCertificate;
    UA_MessageSecurityMode securityMode;
    UA_String securityPolicyUri;
    size_t userIdentityTokensSize;
    UA_UserTokenPolicy *userIdentityTokens;
    UA_String transportProfileUri;
    UA_Byte securityLevel;
} UA_EndpointDescription;

#define UA_TYPES_ENDPOINTDESCRIPTION 183
```

5.5.185 CreateMonitoredItemsRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_UInt32 subscriptionId;
    UA_TimestampsToReturn timestampsToReturn;
    size_t itemsToCreateSize;
    UA_MonitoredItemCreateRequest *itemsToCreate;
} UA_CreateMonitoredItemsRequest;
```

```
#define UA_TYPES_CREATEMONITOREDITEMSREQUEST 184
```

5.5.186 ContentFilter

```
typedef struct {
    size_t elementsSize;
    UA_ContentFilterElement *elements;
} UA_ContentFilter;

#define UA_TYPES_CONTENTFILTER 185
```

5.5.187 QueryFirstResponse

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t queryDataSetsSize;
    UA_QueryDataSet *queryDataSets;
    UA_ByteString continuationPoint;
    size_t parsingResultsSize;
    UA_ParsingResult *parsingResults;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
    UA_ContentFilterResult filterResult;
} UA_QueryFirstResponse;

#define UA_TYPES_QUERYFIRSTRESPONSE 186
```

5.5.188 AddNodesRequest

Adds one or more nodes to the server address space.

```
typedef struct {
    UA_RequestHeader requestHeader;
    size_t nodesToAddSize;
    UA_AddNodesItem *nodesToAdd;
} UA_AddNodesRequest;

#define UA_TYPES_ADDNODESREQUEST 187
```

5.5.189 BrowseRequest

Browse the references for one or more nodes from the server address space.

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_ViewDescription view;
    UA_UInt32 requestedMaxReferencesPerNode;
    size_t nodesToBrowseSize;
    UA_BrowseDescription *nodesToBrowse;
} UA_BrowseRequest;

#define UA_TYPES_BROWSEREQUEST 188
```

5.5.190 BrowseResult

The result of a browse operation.

```
typedef struct {
    UA_StatusCode statusCode;
    UA_ByteString continuationPoint;
    size_t referencesSize;
    UA_ReferenceDescription *references;
} UA_BrowseResult;

#define UA_TYPES_BROWSERESULT 189
```

5.5.191 RegisterServer2Request

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_RegisteredServer server;
    size_t discoveryConfigurationSize;
    UA_ExtensionObject *discoveryConfiguration;
} UA_RegisterServer2Request;

#define UA_TYPES_REGISTERSERVER2REQUEST 190
```

5.5.192 CreateSessionRequest

Creates a new session with the server.

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_ApplicationDescription clientDescription;
    UA_String serverUri;
    UA_String endpointUrl;
    UA_String sessionName;
    UA_ByteString clientNonce;
    UA_ByteString clientCertificate;
    UA_Double requestedSessionTimeout;
    UA_UInt32 maxResponseMessageSize;
} UA_CreateSessionRequest;

#define UA_TYPES_CREATESESSIONREQUEST 191
```

5.5.193 EventFilter

```
typedef struct {
    size_t selectClausesSize;
    UA_SimpleAttributeOperand *selectClauses;
    UA_ContentFilter whereClause;
} UA_EventFilter;

#define UA_TYPES_EVENTFILTER 192
```

5.5.194 GetEndpointsResponse

Gets the endpoints used by the server.

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t endpointsSize;
    UA_EndpointDescription *endpoints;
} UA_GetEndpointsResponse;

#define UA_TYPES_GETENDPOINTSRESPONSE 193
```

5.5.195 FindServersResponse

Finds the servers known to the discovery server.

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t serversSize;
    UA_ApplicationDescription *servers;
} UA_FindServersResponse;

#define UA_TYPES_FINDSERVERSRESPONSE 194
```

5.5.196 BrowseNextResponse

Continues one or more browse operations.

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_BrowseResult *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_BrowseNextResponse;

#define UA_TYPES_BROWSENEXTRESPONSE 195
```

5.5.197 BrowseResponse

Browse the references for one or more nodes from the server address space.

```
typedef struct {
    UA_ResponseHeader responseHeader;
    size_t resultsSize;
    UA_BrowseResult *results;
    size_t diagnosticInfosSize;
    UA_DiagnosticInfo *diagnosticInfos;
} UA_BrowseResponse;

#define UA_TYPES_BROWSERESPONSE 196
```

5.5.198 CreateSessionResponse

Creates a new session with the server.

```
typedef struct {
    UA_ResponseHeader responseHeader;
    UA_NodeId sessionId;
```

```
    UA_NodeId authenticationToken;
    UA_Double revisedSessionTimeout;
    UA_ByteString serverNonce;
    UA_ByteString serverCertificate;
    size_t serverEndpointsSize;
    UA_EndpointDescription *serverEndpoints;
    size_t serverSoftwareCertificatesSize;
    UA_SignedSoftwareCertificate *serverSoftwareCertificates;
    UA_SignatureData serverSignature;
    UA_UInt32 maxRequestMessageSize;
} UA_CreateSessionResponse;

#define UA_TYPES_CREATESESSIONRESPONSE 197
```

5.5.199 QueryFirstRequest

```
typedef struct {
    UA_RequestHeader requestHeader;
    UA_ViewDescription view;
    size_t nodeTypesSize;
    UA_NodeTypeDescription *nodeTypes;
    UA_ContentFilter filter;
    UA_UInt32 maxDataSetsToReturn;
    UA_UInt32 maxReferencesToReturn;
} UA_QueryFirstRequest;

#define UA_TYPES_QUERYFIRSTREQUEST 198
```

5.5.200 UtcTime

A date/time value specified in Universal Coordinated Time (UTC).

```
typedef UA_DateTime UA_UtcTime;

#define UA_TYPES_UTCTIME UA_TYPES_DATETIME
```

5.5.201 LocaleId

An identifier for a user locale.

```
typedef UA_String UA_LocaleId;

#define UA_TYPES_LOCALEID UA_TYPES_STRING
```

5.5.202 Duration

A period of time measured in milliseconds.

```
typedef UA_Double UA_Duration;

#define UA_TYPES_DURATION UA_TYPES_DOUBLE
```

5.6 Deprecated Data Types API

The following definitions are deprecated and will be removed in future releases of open62541.

```c
typedef struct {
    UA_StatusCode code;       /* The numeric value of the StatusCode */
    const char* name;         /* The symbolic name */
    const char* explanation; /* Short message explaining the StatusCode */
} UA_StatusCodeDescription;

extern const UA_StatusCodeDescription statusCodeExplanation_default;

UA_DEPRECATED static UA_INLINE const UA_StatusCodeDescription *
UA_StatusCode_description(UA_StatusCode code) {
    return &statusCodeExplanation_default;
}

UA_DEPRECATED static UA_INLINE const char *
UA_StatusCode_explanation(UA_StatusCode code) {
    return statusCodeExplanation_default.name;
}

UA_DEPRECATED UA_String
UA_DateTime_toString(UA_DateTime t);

/* The old DateTime conversion macros */
UA_DEPRECATED static UA_INLINE double
deprecatedDateTimeMultiple(double multiple) {
    return multiple;
}

#define UA_USEC_TO_DATETIME deprecatedDateTimeMultiple((UA_Double)UA_DATETIME_USEC)
#define UA_MSEC_TO_DATETIME deprecatedDateTimeMultiple((UA_Double)UA_DATETIME_MSEC)
#define UA_SEC_TO_DATETIME deprecatedDateTimeMultiple((UA_Double)UA_DATETIME_SEC)
#define UA_DATETIME_TO_USEC deprecatedDateTimeMultiple(1.0 / ((UA_Double)UA_
↪DATETIME_USEC))
#define UA_DATETIME_TO_MSEC deprecatedDateTimeMultiple(1.0 / ((UA_Double)UA_
↪DATETIME_MSEC))
#define UA_DATETIME_TO_SEC deprecatedDateTimeMultiple(1.0 / ((UA_Double)UA_
↪DATETIME_SEC))
```

Services

In OPC UA, all communication is based on service calls, each consisting of a request and a response message. These messages are defined as data structures with a binary encoding and listed in *Generated Data Type Definitions*. Since all Services are pre-defined in the standard, they cannot be modified by the user. But you can use the *Call* service to invoke user-defined methods on the server.

The following service signatures are internal and *not visible to users*. Still, we present them here for an overview of the capabilities of OPC UA. Please refer to the *Client* and *Server* API where the services are exposed to end users. Please see part 4 of the OPC UA standard for the authoritative definition of the service and their behaviour.

Most services take as input the server, the current session and pointers to the request and response structures. Possible error codes are returned as part of the response.

```
typedef void (*UA_Service)(UA_Server*, UA_Session*,
                           const void *request, void *response);

typedef UA_StatusCode (*UA_InSituService)(UA_Server*, UA_Session*, UA_
→MessageContext *mc,
                                const void *request, UA_ResponseHeader␣
→*rh);
```

6.1 Discovery Service Set

This Service Set defines Services used to discover the Endpoints implemented by a Server and to read the security configuration for those Endpoints.

6.1.1 FindServers Service

Returns the Servers known to a Server or Discovery Server. The Client may reduce the number of results returned by specifying filter criteria

```
void Service_FindServers(UA_Server *server, UA_Session *session,
                         const UA_FindServersRequest *request,
                         UA_FindServersResponse *response);
```

6.1.2 GetEndpoints Service

Returns the Endpoints supported by a Server and all of the configuration information required to establish a SecureChannel and a Session.

```
void Service_GetEndpoints(UA_Server *server, UA_Session *session,
                          const UA_GetEndpointsRequest *request,
                          UA_GetEndpointsResponse *response);

#ifdef UA_ENABLE_DISCOVERY

# ifdef UA_ENABLE_DISCOVERY_MULTICAST
```

6.1.3 FindServersOnNetwork Service

Returns the Servers known to a Discovery Server. Unlike FindServer, this Service is only implemented by Discovery Servers. It additionally Returns servery which may have been detected trough Multicast

```
void Service_FindServersOnNetwork(UA_Server *server, UA_Session *session,
                                  const UA_FindServersOnNetworkRequest *request,
                                  UA_FindServersOnNetworkResponse *response);

# endif /* UA_ENABLE_DISCOVERY_MULTICAST */
```

6.1.4 RegisterServer

Registers a remote server in the local discovery service.

```
void Service_RegisterServer(UA_Server *server, UA_Session *session,
                            const UA_RegisterServerRequest *request,
                            UA_RegisterServerResponse *response);
```

6.1.5 RegisterServer2

This Service allows a Server to register its DiscoveryUrls and capabilities with a Discovery Server. It extends the registration information from RegisterServer with information necessary for FindServersOnNetwork.

```
void Service_RegisterServer2(UA_Server *server, UA_Session *session,
                             const UA_RegisterServer2Request *request,
                             UA_RegisterServer2Response *response);

#endif /* UA_ENABLE_DISCOVERY */
```

6.2 SecureChannel Service Set

This Service Set defines Services used to open a communication channel that ensures the confidentiality and Integrity of all Messages exchanged with the Server.

6.2.1 OpenSecureChannel Service

Open or renew a SecureChannel that can be used to ensure Confidentiality and Integrity for Message exchange during a Session.

```
void Service_OpenSecureChannel(UA_Server *server, UA_SecureChannel* channel,
                               const UA_OpenSecureChannelRequest *request,
                               UA_OpenSecureChannelResponse *response);
```

6.2.2 CloseSecureChannel Service

Used to terminate a SecureChannel.

```
void Service_CloseSecureChannel(UA_Server *server, UA_SecureChannel *channel);
```

6.3 Session Service Set

This Service Set defines Services for an application layer connection establishment in the context of a Session.

6.3.1 CreateSession Service

Used by an OPC UA Client to create a Session and the Server returns two values which uniquely identify the Session. The first value is the sessionId which is used to identify the Session in the audit logs and in the Server's address space. The second is the authenticationToken which is used to associate an incoming request with a Session.

```
void Service_CreateSession(UA_Server *server, UA_SecureChannel *channel,
                           const UA_CreateSessionRequest *request,
                           UA_CreateSessionResponse *response);
```

6.3.2 ActivateSession

Used by the Client to submit its SoftwareCertificates to the Server for validation and to specify the identity of the user associated with the Session. This Service request shall be issued by the Client before it issues any other Service request after CreateSession. Failure to do so shall cause the Server to close the Session.

```
void Service_ActivateSession(UA_Server *server, UA_SecureChannel *channel,
                             UA_Session *session,
                             const UA_ActivateSessionRequest *request,
                             UA_ActivateSessionResponse *response);
```

6.3.3 CloseSession

Used to terminate a Session.

```
void Service_CloseSession(UA_Server *server, UA_Session *session,
                          const UA_CloseSessionRequest *request,
                          UA_CloseSessionResponse *response);
```

6.3.4 Cancel Service

Used to cancel outstanding Service requests. Successfully cancelled service requests shall respond with Bad_RequestCancelledByClient.

```
/* Not Implemented */
```

6.4 NodeManagement Service Set

This Service Set defines Services to add and delete AddressSpace Nodes and References between them. All added Nodes continue to exist in the AddressSpace even if the Client that created them disconnects from the Server.

6.4.1 AddNodes Service

Used to add one or more Nodes into the AddressSpace hierarchy.

```
void Service_AddNodes(UA_Server *server, UA_Session *session,
                      const UA_AddNodesRequest *request,
                      UA_AddNodesResponse *response);
```

6.4.2 AddReferences Service

Used to add one or more References to one or more Nodes.

```
void Service_AddReferences(UA_Server *server, UA_Session *session,
                           const UA_AddReferencesRequest *request,
                           UA_AddReferencesResponse *response);
```

6.4.3 DeleteNodes Service

Used to delete one or more Nodes from the AddressSpace.

```
void Service_DeleteNodes(UA_Server *server, UA_Session *session,
                         const UA_DeleteNodesRequest *request,
                         UA_DeleteNodesResponse *response);
```

6.4.4 DeleteReferences

Used to delete one or more References of a Node.

```
void Service_DeleteReferences(UA_Server *server, UA_Session *session,
                              const UA_DeleteReferencesRequest *request,
                              UA_DeleteReferencesResponse *response);
```

6.5 View Service Set

Clients use the browse Services of the View Service Set to navigate through the AddressSpace or through a View which is a subset of the AddressSpace.

6.5.1 Browse Service

Used to discover the References of a specified Node. The browse can be further limited by the use of a View. This Browse Service also supports a primitive filtering capability.

```
void Service_Browse(UA_Server *server, UA_Session *session,
                    const UA_BrowseRequest *request,
                    UA_BrowseResponse *response);
```

6.5.2 BrowseNext Service

Used to request the next set of Browse or BrowseNext response information that is too large to be sent in a single response. "Too large" in this context means that the Server is not able to return a larger response or that the number of results to return exceeds the maximum number of results to return that was specified by the Client in the original Browse request.

```
void Service_BrowseNext(UA_Server *server, UA_Session *session,
                        const UA_BrowseNextRequest *request,
                        UA_BrowseNextResponse *response);
```

6.5.3 TranslateBrowsePathsToNodeIds Service

Used to translate textual node paths to their respective ids.

```
void Service_TranslateBrowsePathsToNodeIds(UA_Server *server, UA_Session *session,
            const UA_TranslateBrowsePathsToNodeIdsRequest *request,
            UA_TranslateBrowsePathsToNodeIdsResponse *response);
```

6.5.4 RegisterNodes Service

Used by Clients to register the Nodes that they know they will access repeatedly (e.g. Write, Call). It allows Servers to set up anything needed so that the access operations will be more efficient.

```
void Service_RegisterNodes(UA_Server *server, UA_Session *session,
                           const UA_RegisterNodesRequest *request,
                           UA_RegisterNodesResponse *response);
```

6.5.5 UnregisterNodes Service

This Service is used to unregister NodeIds that have been obtained via the RegisterNodes service.

```
void Service_UnregisterNodes(UA_Server *server, UA_Session *session,
                             const UA_UnregisterNodesRequest *request,
                             UA_UnregisterNodesResponse *response);
```

6.6 Query Service Set

This Service Set is used to issue a Query to a Server. OPC UA Query is generic in that it provides an underlying storage mechanism independent Query capability that can be used to access a wide variety of OPC UA data stores and information management systems. OPC UA Query permits a Client to access data maintained by a Server without any knowledge of the logical schema used for internal storage of the data. Knowledge of the AddressSpace is sufficient.

6.6.1 QueryFirst Service

This Service is used to issue a Query request to the Server.

```
/* Not Implemented */
```

6.6.2 QueryNext Service

This Service is used to request the next set of QueryFirst or QueryNext response information that is too large to be sent in a single response.

```
/* Not Impelemented */
```

6.7 Attribute Service Set

This Service Set provides Services to access Attributes that are part of Nodes.

6.7.1 Read Service

Used to read attributes of nodes. For constructed attribute values whose elements are indexed, such as an array, this Service allows Clients to read the entire set of indexed values as a composite, to read individual elements or to read ranges of elements of the composite.

```
UA_StatusCode Service_Read(UA_Server *server, UA_Session *session, UA_
→MessageContext *mc,
                          const UA_ReadRequest *request, UA_ResponseHeader␣
→*responseHeader);
```

6.7.2 Write Service

Used to write attributes of nodes. For constructed attribute values whose elements are indexed, such as an array, this Service allows Clients to write the entire set of indexed values as a composite, to write individual elements or to write ranges of elements of the composite.

```
void Service_Write(UA_Server *server, UA_Session *session,
                   const UA_WriteRequest *request,
                   UA_WriteResponse *response);
```

6.7.3 HistoryRead Service

Used to read historical values or Events of one or more Nodes. Servers may make historical values available to Clients using this Service, although the historical values themselves are not visible in the AddressSpace.

```
/* Not Implemented */
```

6.7.4 HistoryUpdate Service

Used to update historical values or Events of one or more Nodes. Several request parameters indicate how the Server is to update the historical value or Event. Valid actions are Insert, Replace or Delete.

```
/* Not Implemented */
```

6.8 Method Service Set

The Method Service Set defines the means to invoke methods. A method shall be a component of an Object. See the section on *MethodNodes* for more information.

6.8.1 Call Service

Used to call (invoke) a methods. Each method call is invoked within the context of an existing Session. If the Session is terminated, the results of the method's execution cannot be returned to the Client and are discarded.

```
void Service_Call(UA_Server *server, UA_Session *session,
                  const UA_CallRequest *request,
                  UA_CallResponse *response);
```

6.9 MonitoredItem Service Set

Clients define MonitoredItems to subscribe to data and Events. Each MonitoredItem identifies the item to be monitored and the Subscription to use to send Notifications. The item to be monitored may be any Node Attribute.

6.9.1 CreateMonitoredItems Service

Used to create and add one or more MonitoredItems to a Subscription. A MonitoredItem is deleted automatically by the Server when the Subscription is deleted. Deleting a MonitoredItem causes its entire set of triggered item links to be deleted, but has no effect on the MonitoredItems referenced by the triggered items.

```
void Service_CreateMonitoredItems(UA_Server *server, UA_Session *session,
                                  const UA_CreateMonitoredItemsRequest *request,
                                  UA_CreateMonitoredItemsResponse *response);
```

6.9.2 DeleteMonitoredItems Service

Used to remove one or more MonitoredItems of a Subscription. When a MonitoredItem is deleted, its triggered item links are also deleted.

```
void Service_DeleteMonitoredItems(UA_Server *server, UA_Session *session,
                                  const UA_DeleteMonitoredItemsRequest *request,
                                  UA_DeleteMonitoredItemsResponse *response);
```

6.9.3 ModifyMonitoredItems Service

Used to modify MonitoredItems of a Subscription. Changes to the MonitoredItem settings shall be applied immediately by the Server. They take effect as soon as practical but not later than twice the new revisedSamplingInterval.

Illegal request values for parameters that can be revised do not generate errors. Instead the server will choose default values and indicate them in the corresponding revised parameter.

```
void Service_ModifyMonitoredItems(UA_Server *server, UA_Session *session,
                                  const UA_ModifyMonitoredItemsRequest *request,
                                  UA_ModifyMonitoredItemsResponse *response);
```

6.9.4 SetMonitoringMode Service

Used to set the monitoring mode for one or more MonitoredItems of a Subscription.

```
void Service_SetMonitoringMode(UA_Server *server, UA_Session *session,
                               const UA_SetMonitoringModeRequest *request,
                               UA_SetMonitoringModeResponse *response);
```

6.9.5 SetTriggering Service

Used to create and delete triggering links for a triggering item.

```
/* Not Implemented */
```

6.10 Subscription Service Set

Subscriptions are used to report Notifications to the Client.

6.10.1 CreateSubscription Service

Used to create a Subscription. Subscriptions monitor a set of MonitoredItems for Notifications and return them to the Client in response to Publish requests.

```
void Service_CreateSubscription(UA_Server *server, UA_Session *session,
                                const UA_CreateSubscriptionRequest *request,
                                UA_CreateSubscriptionResponse *response);
```

6.10.2 ModifySubscription Service

Used to modify a Subscription.

```
void Service_ModifySubscription(UA_Server *server, UA_Session *session,
                                const UA_ModifySubscriptionRequest *request,
                                UA_ModifySubscriptionResponse *response);
```

6.10.3 SetPublishingMode Service

Used to enable sending of Notifications on one or more Subscriptions.

```
void Service_SetPublishingMode(UA_Server *server, UA_Session *session,
                               const UA_SetPublishingModeRequest *request,
                               UA_SetPublishingModeResponse *response);
```

6.10.4 Publish Service

Used for two purposes. First, it is used to acknowledge the receipt of NotificationMessages for one or more Subscriptions. Second, it is used to request the Server to return a NotificationMessage or a keep-alive Message.

Note that the service signature is an exception and does not contain a pointer to a PublishResponse. That is because the service queues up publish requests internally and sends responses asynchronously based on timeouts.

```
void Service_Publish(UA_Server *server, UA_Session *session,
                     const UA_PublishRequest *request, UA_UInt32 requestId);
```

6.10.5 Republish Service

Requests the Subscription to republish a NotificationMessage from its retransmission queue.

```
void Service_Republish(UA_Server *server, UA_Session *session,
                       const UA_RepublishRequest *request,
                       UA_RepublishResponse *response);
```

6.10.6 DeleteSubscriptions Service

Invoked to delete one or more Subscriptions that belong to the Client's Session.

```c
void Service_DeleteSubscriptions(UA_Server *server, UA_Session *session,
                                 const UA_DeleteSubscriptionsRequest *request,
                                 UA_DeleteSubscriptionsResponse *response);
```

6.10.7 TransferSubscription Service

Used to transfer a Subscription and its MonitoredItems from one Session to another. For example, a Client may need to reopen a Session and then transfer its Subscriptions to that Session. It may also be used by one Client to take over a Subscription from another Client by transferring the Subscription to its Session.

```c
/* Not Implemented */
```

Information Modelling

Information modelling in OPC UA combines concepts from object-orientation and semantic modelling. At the core, an OPC UA information model is a graph made up of

- Nodes: There are eight possible Node types (variable, object, method, . . .)

- References: Typed and directed relations between two nodes

Every node is identified by a unique (within the server) *NodeId*. Reference are triples of the form `(source-nodeid, referencetype-nodeid, target-nodeid)`. An example reference between nodes is a `hasTypeDefinition` reference between a Variable and its VariableType. Some ReferenceTypes are *hierarchic* and must not form *directed loops*. See the section on *ReferenceTypes* for more details on possible references and their semantics.

Warning!! The structures defined in this section are only relevant for the developers of custom Nodestores. The interaction with the information model is possible only via the OPC UA *Services*. So the following sections are purely informational so that users may have a clear mental model of the underlying representation.

7.1 Base Node Attributes

Nodes contain attributes according to their node type. The base node attributes are common to all node types. In the OPC UA *Services*, attributes are referred to via the *NodeId* of the containing node and an integer *Attribute Id*.

Internally, open62541 uses `UA_Node` in places where the exact node type is not known or not important. The `nodeClass` attribute is used to ensure the correctness of casting from `UA_Node` to a specific node type.

```
/* List of reference targets with the same reference type and direction */
typedef struct {
    UA_NodeId referenceTypeId;
    UA_Boolean isInverse;
    size_t targetIdsSize;
    UA_ExpandedNodeId *targetIds;
} UA_NodeReferenceKind;

#define UA_NODE_BASEATTRIBUTES                   \
    UA_NodeId nodeId;                            \
    UA_NodeClass nodeClass;                      \
    UA_QualifiedName browseName;                 \
    UA_LocalizedText displayName;                \
```

```
    UA_LocalizedText description;              \
    UA_UInt32 writeMask;                       \
    size_t referencesSize;                     \
    UA_NodeReferenceKind *references;          \
                                               \
    /* Members specific to open62541 */        \
    void *context;

typedef struct {
    UA_NODE_BASEATTRIBUTES
} UA_Node;
```

7.2 VariableNode

Variables store values in a *DataValue* together with metadata for introspection. Most notably, the attributes data type, value rank and array dimensions constrain the possible values the variable can take on.

Variables come in two flavours: properties and datavariables. Properties are related to a parent with a `hasProperty` reference and may not have child nodes themselves. Datavariables may contain properties (`hasProperty`) and also datavariables (`hasComponents`).

All variables are instances of some *VariableTypeNode* in return constraining the possible data type, value rank and array dimensions attributes.

7.2.1 Data Type

The (scalar) data type of the variable is constrained to be of a specific type or one of its children in the type hierarchy. The data type is given as a NodeId pointing to a *DataTypeNode* in the type hierarchy. See the Section *DataTypeNode* for more details.

If the data type attribute points to `UInt32`, then the value attribute must be of that exact type since `UInt32` does not have children in the type hierarchy. If the data type attribute points `Number`, then the type of the value attribute may still be `UInt32`, but also `Float` or `Byte`.

Consistency between the data type attribute in the variable and its *VariableTypeNode* is ensured.

7.2.2 Value Rank

This attribute indicates whether the value attribute of the variable is an array and how many dimensions the array has. It may have the following values:

- `n >= 1`: the value is an array with the specified number of dimensions
- `n = 0`: the value is an array with one or more dimensions
- `n = -1`: the value is a scalar
- `n = -2`: the value can be a scalar or an array with any number of dimensions
- `n = -3`: the value can be a scalar or a one dimensional array

Consistency between the value rank attribute in the variable and its *VariableTypeNode* is ensured.

7.2.3 Array Dimensions

If the value rank permits the value to be a (multi-dimensional) array, the exact length in each dimensions can be further constrained with this attribute.

- For positive lengths, the variable value is guaranteed to be of the same length in this dimension.

- The dimension length zero is a wildcard and the actual value may have any length in this dimension.

Consistency between the array dimensions attribute in the variable and its *VariableTypeNode* is ensured.

```c
/* Indicates whether a variable contains data inline or whether it points to an
 * external data source */
typedef enum {
    UA_VALUESOURCE_DATA,
    UA_VALUESOURCE_DATASOURCE
} UA_ValueSource;

#define UA_NODE_VARIABLEATTRIBUTES                                            \
    /* Constraints on possible values */                                     \
    UA_NodeId dataType;                                                      \
    UA_Int32 valueRank;                                                      \
    size_t arrayDimensionsSize;                                             \
    UA_UInt32 *arrayDimensions;                                             \
                                                                            \
    /* The current value */                                                 \
    UA_ValueSource valueSource;                                             \
    union {                                                                 \
        struct {                                                            \
            UA_DataValue value;                                             \
            UA_ValueCallback callback;                                      \
        } data;                                                             \
        UA_DataSource dataSource;                                           \
    } value;

typedef struct {
    UA_NODE_BASEATTRIBUTES
    UA_NODE_VARIABLEATTRIBUTES
    UA_Byte accessLevel;
    UA_Double minimumSamplingInterval;
    UA_Boolean historizing; /* currently unsupported */
} UA_VariableNode;
```

7.3 VariableTypeNode

VariableTypes are used to provide type definitions for variables. VariableTypes constrain the data type, value rank and array dimensions attributes of variable instances. Furthermore, instantiating from a specific variable type may provide semantic information. For example, an instance from `MotorTemperatureVariableType` is more meaningful than a float variable instantiated from `BaseDataVariable`.

```c
typedef struct {
    UA_NODE_BASEATTRIBUTES
    UA_NODE_VARIABLEATTRIBUTES
    UA_Boolean isAbstract;

    /* Members specific to open62541 */
    UA_NodeTypeLifecycle lifecycle;
} UA_VariableTypeNode;
```

7.4 MethodNode

Methods define callable functions and are invoked using the *Call* service. MethodNodes may have special properties (variable childen with a `hasProperty` reference) with the *QualifiedName* (0, `"InputArguments"`) and (0, `"OutputArguments"`). The input and output arguments are both described via an array of

UA_Argument. While the Call service uses a generic array of *Variant* for input and output, the actual argument values are checked to match the signature of the MethodNode.

Note that the same MethodNode may be referenced from several objects (and object types). For this, the NodeId of the method *and of the object providing context* is part of a Call request message.

```
typedef struct {
    UA_NODE_BASEATTRIBUTES
    UA_Boolean executable;

    /* Members specific to open62541 */
    UA_MethodCallback method;
} UA_MethodNode;
```

7.5 ObjectNode

Objects are used to represent systems, system components, real-world objects and software objects. Objects are instances of an *object type* and may contain variables, methods and further objects.

```
typedef struct {
    UA_NODE_BASEATTRIBUTES
    UA_Byte eventNotifier;
} UA_ObjectNode;
```

7.6 ObjectTypeNode

ObjectTypes provide definitions for Objects. Abstract objects cannot be instantiated. See *Node Lifecycle: Constructors, Destructors and Node Contexts* for the use of constructor and destructor callbacks.

```
typedef struct {
    UA_NODE_BASEATTRIBUTES
    UA_Boolean isAbstract;

    /* Members specific to open62541 */
    UA_NodeTypeLifecycle lifecycle;
} UA_ObjectTypeNode;
```

7.7 ReferenceTypeNode

Each reference between two nodes is typed with a ReferenceType that gives meaning to the relation. The OPC UA standard defines a set of ReferenceTypes as a mandatory part of OPC UA information models.

- Abstract ReferenceTypes cannot be used in actual references and are only used to structure the ReferenceTypes hierarchy

- Symmetric references have the same meaning from the perspective of the source and target node

The figure below shows the hierarchy of the standard ReferenceTypes (arrows indicate a `hasSubType` relation). Refer to Part 3 of the OPC UA specification for the full semantics of each ReferenceType.

The ReferenceType hierarchy can be extended with user-defined ReferenceTypes. Many Companion Specifications for OPC UA define new ReferenceTypes to be used in their domain of interest.

For the following example of custom ReferenceTypes, we attempt to model the structure of a technical system. For this, we introduce two custom ReferenceTypes. First, the hierarchical `contains` ReferenceType indicates that a system (represented by an OPC UA object) contains a component (or subsystem). This gives rise to a tree-structure of containment relations. For example, the motor (object) is contained in the car and the crankshaft is contained in the motor. Second, the symmetric `connectedTo` ReferenceType indicates that two components are connected. For example, the motor's crankshaft is connected to the gear box. Connections are independent of the containment hierarchy and can induce a general graph-structure. Further subtypes of `connectedTo` could be used to differentiate between physical, electrical and information related connections. A client can then learn the layout of a (physical) system represented in an OPC UA information model based on a common understanding of just two custom reference types.

```
typedef struct {
    UA_NODE_BASEATTRIBUTES
    UA_Boolean isAbstract;
    UA_Boolean symmetric;
    UA_LocalizedText inverseName;
} UA_ReferenceTypeNode;
```

7.8 DataTypeNode

DataTypes represent simple and structured data types. DataTypes may contain arrays. But they always describe the structure of a single instance. In open62541, DataTypeNodes in the information model hierarchy are matched to `UA_DataType` type descriptions for *Generic Type Handling* via their NodeId.

Abstract DataTypes (e.g. `Number`) cannot be the type of actual values. They are used to constrain values to possible child DataTypes (e.g. `UInt32`).

```
typedef struct {
    UA_NODE_BASEATTRIBUTES
    UA_Boolean isAbstract;
} UA_DataTypeNode;
```

7.9 ViewNode

Each View defines a subset of the Nodes in the AddressSpace. Views can be used when browsing an information model to focus on a subset of nodes and references only. ViewNodes can be created and be interacted with. But their use in the *Browse* service is currently unsupported in open62541.

```
typedef struct {
    UA_NODE_BASEATTRIBUTES
    UA_Byte eventNotifier;
    UA_Boolean containsNoLoops;
} UA_ViewNode;
```

7.10 Nodestore Plugin API

The following definitions are used for implementing custom node storage backends. **Most users will want to use the default nodestore and don't need to work with the nodestore API**.

Outside of custom nodestore implementations, users should not manually edit nodes. Please use the OPC UA services for that. Otherwise, all consistency checks are omitted. This can crash the application eventually.

```
typedef void (*UA_NodestoreVisitor)(void *visitorContext, const UA_Node *node);

typedef struct {
    /* Nodestore context and lifecycle */
    void *context;
    void (*deleteNodestore)(void *nodestoreContext);

    /* For non-multithreaded access, some nodestores allow that nodes are edited
     * without a copy/replace. This is not possible when the node is only an
     * intermediate representation and stored e.g. in a database backend. */
    UA_Boolean inPlaceEditAllowed;

    /* The following definitions are used to create empty nodes of the different
     * node types. The memory is managed by the nodestore. Therefore, the node
     * has to be removed via a special deleteNode function. (If the new node is
     * not added to the nodestore.) */
    UA_Node * (*newNode)(void *nodestoreContext, UA_NodeClass nodeClass);

    void (*deleteNode)(void *nodestoreContext, UA_Node *node);

    /* ``Get`` returns a pointer to an immutable node. ``Release`` indicates
     * that the pointer is no longer accessed afterwards. */

    const UA_Node * (*getNode)(void *nodestoreContext, const UA_NodeId *nodeId);

    void (*releaseNode)(void *nodestoreContext, const UA_Node *node);

    /* Returns an editable copy of a node (needs to be deleted with the
     * deleteNode function or inserted / replaced into the nodestore). */
    UA_StatusCode (*getNodeCopy)(void *nodestoreContext, const UA_NodeId *nodeId,
                                 UA_Node **outNode);

    /* Inserts a new node into the nodestore. If the NodeId is zero, then a
     * fresh numeric NodeId is assigned. If insertion fails, the node is
     * deleted. */
    UA_StatusCode (*insertNode)(void *nodestoreContext, UA_Node *node,
                                UA_NodeId *addedNodeId);

    /* To replace a node, get an editable copy of the node, edit and replace
```

```c
     * with this function. If the node was already replaced since the copy was
     * made, UA_STATUSCODE_BADINTERNALERROR is returned. If the NodeId is not
     * found, UA_STATUSCODE_BADNODEIDUNKNOWN is returned. In both error cases,
     * the editable node is deleted. */
    UA_StatusCode (*replaceNode)(void *nodestoreContext, UA_Node *node);

    /* Removes a node from the nodestore. */
    UA_StatusCode (*removeNode)(void *nodestoreContext, const UA_NodeId *nodeId);

    /* Execute a callback for every node in the nodestore. */
    void (*iterate)(void *nodestoreContext, void* visitorContext,
                    UA_NodestoreVisitor visitor);
} UA_Nodestore;
```

The following methods specialize internally for the different node classes (distinguished by the nodeClass member)

```c
/* Attributes must be of a matching type (VariableAttributes, ObjectAttributes,
 * and so on). The attributes are copied. Note that the attributes structs do
 * not contain NodeId, NodeClass and BrowseName. The NodeClass of the node needs
 * to be correctly set before calling this method. UA_Node_deleteMembers is
 * called on the node when an error occurs internally. */
UA_StatusCode
UA_Node_setAttributes(UA_Node *node, const void *attributes,
                      const UA_DataType *attributeType);

/* Reset the destination node and copy the content of the source */
UA_StatusCode
UA_Node_copy(const UA_Node *src, UA_Node *dst);

/* Allocate new node and copy the values from src */
UA_Node *
UA_Node_copy_alloc(const UA_Node *src);

/* Add a single reference to the node */
UA_StatusCode
UA_Node_addReference(UA_Node *node, const UA_AddReferencesItem *item);

/* Delete a single reference from the node */
UA_StatusCode
UA_Node_deleteReference(UA_Node *node, const UA_DeleteReferencesItem *item);

/* Delete all references of the node */
void
UA_Node_deleteReferences(UA_Node *node);

/* Remove all malloc'ed members of the node */
void
UA_Node_deleteMembers(UA_Node *node);
```

Server

8.1 Server Configuration

The configuration structure is passed to the server during initialization. The server expects that the configuration is not modified during runtime. Currently, only one server can use a configuration at a time. During shutdown, the server will clean up the parts of the configuration that are modified at runtime through the provided API.

Examples for configurations are provided in the `/plugins` folder. The usual usage is as follows:

1. Create a server configuration with default settings as a starting point

2. Modifiy the configuration, e.g. by adding a server certificate

3. Instantiate a server with it

4. After shutdown of the server, clean up the configuration (free memory)

The *Tutorials* provide a good starting point for this.

```c
typedef struct {
    UA_UInt32 min;
    UA_UInt32 max;
} UA_UInt32Range;

typedef struct {
    UA_Duration min;
    UA_Duration max;
} UA_DurationRange;

struct UA_ServerConfig {
    UA_UInt16 nThreads; /* only if multithreading is enabled */
    UA_Logger logger;

    /* Server Description */
    UA_BuildInfo buildInfo;
    UA_ApplicationDescription applicationDescription;
    UA_ByteString serverCertificate;

    /* MDNS Discovery */
#ifdef UA_ENABLE_DISCOVERY
    UA_String mdnsServerName;
```

(continues on next page)

```
    size_t serverCapabilitiesSize;
    UA_String *serverCapabilities;
#endif

    /* Custom DataTypes */
    size_t customDataTypesSize;
    UA_DataType *customDataTypes;
```

Note: See the section on *Generic Type Handling*. Examples for working with custom data types are provided in /examples/custom_datatype/.

```
/* Nodestore */
UA_Nodestore nodestore;

/* Networking */
size_t networkLayersSize;
UA_ServerNetworkLayer *networkLayers;
UA_String customHostname;

/* Available endpoints */
size_t endpointsSize;
UA_Endpoint *endpoints;

/* Node Lifecycle callbacks */
UA_GlobalNodeLifecycle nodeLifecycle;
```

Note: See the section for *node lifecycle handling*.

```
/* Access Control */
UA_AccessControl accessControl;
```

Note: See the section for *access-control handling*.

```
    /* Certificate Verification */
    UA_CertificateVerification certificateVerification;

    /* Limits for SecureChannels */
    UA_UInt16 maxSecureChannels;
    UA_UInt32 maxSecurityTokenLifetime; /* in ms */

    /* Limits for Sessions */
    UA_UInt16 maxSessions;
    UA_Double maxSessionTimeout; /* in ms */

    /* Operation limits */
    UA_UInt32 maxNodesPerRead;
    UA_UInt32 maxNodesPerWrite;
    UA_UInt32 maxNodesPerMethodCall;
    UA_UInt32 maxNodesPerBrowse;
    UA_UInt32 maxNodesPerRegisterNodes;
    UA_UInt32 maxNodesPerTranslateBrowsePathsToNodeIds;
    UA_UInt32 maxNodesPerNodeManagement;
    UA_UInt32 maxMonitoredItemsPerCall;
```

```
    /* Limits for Requests */
    UA_UInt32 maxReferencesPerNode;

    /* Limits for Subscriptions */
    UA_UInt32 maxSubscriptionsPerSession;
    UA_DurationRange publishingIntervalLimits; /* in ms (must not be less than 5)␣
↪*/
    UA_UInt32Range lifeTimeCountLimits;
    UA_UInt32Range keepAliveCountLimits;
    UA_UInt32 maxNotificationsPerPublish;
    UA_UInt32 maxRetransmissionQueueSize; /* 0 -> unlimited size */

    /* Limits for MonitoredItems */
    UA_UInt32 maxMonitoredItemsPerSubscription;
    UA_DurationRange samplingIntervalLimits; /* in ms (must not be less than 5) */
    UA_UInt32Range queueSizeLimits; /* Negotiated with the client */

    /* Limits for PublishRequests */
    UA_UInt32 maxPublishReqPerSession;

    /* Discovery */
#ifdef UA_ENABLE_DISCOVERY
    /* Timeout in seconds when to automatically remove a registered server from
     * the list, if it doesn't re-register within the given time frame. A value
     * of 0 disables automatic removal. Default is 60 Minutes (60*60). Must be
     * bigger than 10 seconds, because cleanup is only triggered approximately
     * ervery 10 seconds. The server will still be removed depending on the
     * state of the semaphore file. */
    UA_UInt32 discoveryCleanupTimeout;
#endif
};
```

8.2 Server Lifecycle

```
UA_Server * UA_Server_new(const UA_ServerConfig *config);
void UA_Server_delete(UA_Server *server);

/* Runs the main loop of the server. In each iteration, this calls into the
 * networklayers to see if messages have arrived.
 *
 * @param server The server object.
 * @param running The loop is run as long as *running is true.
 *        Otherwise, the server shuts down.
 * @return Returns the statuscode of the UA_Server_run_shutdown method */
UA_StatusCode
UA_Server_run(UA_Server *server, volatile UA_Boolean *running);

/* The prologue part of UA_Server_run (no need to use if you call
 * UA_Server_run) */
UA_StatusCode
UA_Server_run_startup(UA_Server *server);

/* Executes a single iteration of the server's main loop.
 *
 * @param server The server object.
 * @param waitInternal Should we wait for messages in the networklayer?
 *        Otherwise, the timouts for the networklayers are set to zero.
 *        The default max wait time is 50millisec.
```

```
 * @return Returns how long we can wait until the next scheduled
 *         callback (in ms) */
UA_UInt16
UA_Server_run_iterate(UA_Server *server, UA_Boolean waitInternal);

/* The epilogue part of UA_Server_run (no need to use if you call
 * UA_Server_run) */
UA_StatusCode
UA_Server_run_shutdown(UA_Server *server);
```

8.3 Repeated Callbacks

```
typedef void (*UA_ServerCallback)(UA_Server *server, void *data);

/* Add a callback for cyclic repetition to the server.
 *
 * @param server The server object.
 * @param callback The callback that shall be added.
 * @param interval The callback shall be repeatedly executed with the given␣
→interval
 *        (in ms). The interval must be larger than 5ms. The first execution
 *        occurs at now() + interval at the latest.
 * @param callbackId Set to the identifier of the repeated callback . This can be␣
→used to cancel
 *        the callback later on. If the pointer is null, the identifier is not set.
 * @return Upon success, UA_STATUSCODE_GOOD is returned.
 *         An error code otherwise. */
UA_StatusCode
UA_Server_addRepeatedCallback(UA_Server *server, UA_ServerCallback callback,
                              void *data, UA_UInt32 interval, UA_UInt64␣
→*callbackId);

UA_StatusCode
UA_Server_changeRepeatedCallbackInterval(UA_Server *server, UA_UInt64 callbackId,
                                         UA_UInt32 interval);

/* Remove a repeated callback.
 *
 * @param server The server object.
 * @param callbackId The id of the callback that shall be removed.
 * @return Upon success, UA_STATUSCODE_GOOD is returned.
 *         An error code otherwise. */
UA_StatusCode
UA_Server_removeRepeatedCallback(UA_Server *server, UA_UInt64 callbackId);
```

8.4 Reading and Writing Node Attributes

The functions for reading and writing node attributes call the regular read and write service in the background that are also used over the network.

The following attributes cannot be read, since the local "admin" user always has full rights.

- UserWriteMask
- UserAccessLevel
- UserExecutable

```c
/* Read an attribute of a node. The specialized functions below provide a more
 * concise syntax.
 *
 * @param server The server object.
 * @param item ReadValueIds contain the NodeId of the target node, the id of the
 *             attribute to read and (optionally) an index range to read parts
 *             of an array only. See the section on NumericRange for the format
 *             used for array ranges.
 * @param timestamps Which timestamps to return for the attribute.
 * @return Returns a DataValue that contains either an error code, or a variant
 *         with the attribute value and the timestamps. */
UA_DataValue
UA_Server_read(UA_Server *server, const UA_ReadValueId *item,
               UA_TimestampsToReturn timestamps);


/* Don't use this function. There are typed versions for every supported
 * attribute. */
UA_StatusCode
__UA_Server_read(UA_Server *server, const UA_NodeId *nodeId,
                 UA_AttributeId attributeId, void *v);

static UA_INLINE UA_StatusCode
UA_Server_readNodeId(UA_Server *server, const UA_NodeId nodeId,
                     UA_NodeId *outNodeId) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_NODEID, outNodeId);
}

static UA_INLINE UA_StatusCode
UA_Server_readNodeClass(UA_Server *server, const UA_NodeId nodeId,
                        UA_NodeClass *outNodeClass) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_NODECLASS,
                            outNodeClass);
}

static UA_INLINE UA_StatusCode
UA_Server_readBrowseName(UA_Server *server, const UA_NodeId nodeId,
                         UA_QualifiedName *outBrowseName) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_BROWSENAME,
                            outBrowseName);
}

static UA_INLINE UA_StatusCode
UA_Server_readDisplayName(UA_Server *server, const UA_NodeId nodeId,
                          UA_LocalizedText *outDisplayName) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_DISPLAYNAME,
                            outDisplayName);
}

static UA_INLINE UA_StatusCode
UA_Server_readDescription(UA_Server *server, const UA_NodeId nodeId,
                          UA_LocalizedText *outDescription) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_DESCRIPTION,
                            outDescription);
}

static UA_INLINE UA_StatusCode
UA_Server_readWriteMask(UA_Server *server, const UA_NodeId nodeId,
                        UA_UInt32 *outWriteMask) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_WRITEMASK,
                            outWriteMask);
}
```

```
static UA_INLINE UA_StatusCode
UA_Server_readIsAbstract(UA_Server *server, const UA_NodeId nodeId,
                         UA_Boolean *outIsAbstract) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_ISABSTRACT,
                            outIsAbstract);
}

static UA_INLINE UA_StatusCode
UA_Server_readSymmetric(UA_Server *server, const UA_NodeId nodeId,
                        UA_Boolean *outSymmetric) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_SYMMETRIC,
                            outSymmetric);
}

static UA_INLINE UA_StatusCode
UA_Server_readInverseName(UA_Server *server, const UA_NodeId nodeId,
                          UA_LocalizedText *outInverseName) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_INVERSENAME,
                            outInverseName);
}

static UA_INLINE UA_StatusCode
UA_Server_readContainsNoLoop(UA_Server *server, const UA_NodeId nodeId,
                             UA_Boolean *outContainsNoLoops) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_CONTAINSNOLOOPS,
                            outContainsNoLoops);
}

static UA_INLINE UA_StatusCode
UA_Server_readEventNotifier(UA_Server *server, const UA_NodeId nodeId,
                            UA_Byte *outEventNotifier) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_EVENTNOTIFIER,
                            outEventNotifier);
}

static UA_INLINE UA_StatusCode
UA_Server_readValue(UA_Server *server, const UA_NodeId nodeId,
                    UA_Variant *outValue) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_VALUE, outValue);
}

static UA_INLINE UA_StatusCode
UA_Server_readDataType(UA_Server *server, const UA_NodeId nodeId,
                       UA_NodeId *outDataType) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_DATATYPE,
                            outDataType);
}

static UA_INLINE UA_StatusCode
UA_Server_readValueRank(UA_Server *server, const UA_NodeId nodeId,
                        UA_Int32 *outValueRank) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_VALUERANK,
                            outValueRank);
}

/* Returns a variant with an int32 array */
static UA_INLINE UA_StatusCode
UA_Server_readArrayDimensions(UA_Server *server, const UA_NodeId nodeId,
                              UA_Variant *outArrayDimensions) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_ARRAYDIMENSIONS,
                            outArrayDimensions);
```

```
}

static UA_INLINE UA_StatusCode
UA_Server_readAccessLevel(UA_Server *server, const UA_NodeId nodeId,
                          UA_Byte *outAccessLevel) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_ACCESSLEVEL,
                            outAccessLevel);
}

static UA_INLINE UA_StatusCode
UA_Server_readMinimumSamplingInterval(UA_Server *server, const UA_NodeId nodeId,
                                      UA_Double *outMinimumSamplingInterval) {
    return __UA_Server_read(server, &nodeId,
                            UA_ATTRIBUTEID_MINIMUMSAMPLINGINTERVAL,
                            outMinimumSamplingInterval);
}

static UA_INLINE UA_StatusCode
UA_Server_readHistorizing(UA_Server *server, const UA_NodeId nodeId,
                          UA_Boolean *outHistorizing) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_HISTORIZING,
                            outHistorizing);
}

static UA_INLINE UA_StatusCode
UA_Server_readExecutable(UA_Server *server, const UA_NodeId nodeId,
                         UA_Boolean *outExecutable) {
    return __UA_Server_read(server, &nodeId, UA_ATTRIBUTEID_EXECUTABLE,
                            outExecutable);
}
```

The following node attributes cannot be changed once a node has been created:

- NodeClass
- NodeId
- Symmetric
- ContainsNoLoop

The following attributes cannot be written from the server, as they are specific to the different users and set by the access control callback:

- UserWriteMask
- UserAccessLevel
- UserExecutable

Historizing is currently unsupported

```
/* Overwrite an attribute of a node. The specialized functions below provide a
 * more concise syntax.
 *
 * @param server The server object.
 * @param value WriteValues contain the NodeId of the target node, the id of the
 *              attribute to overwritten, the actual value and (optionally) an
 *              index range to replace parts of an array only. of an array only.
 *              See the section on NumericRange for the format used for array
 *              ranges.
 * @return Returns a status code. */
UA_StatusCode
UA_Server_write(UA_Server *server, const UA_WriteValue *value);
```

```c
/* Don't use this function. There are typed versions with no additional
 * overhead. */
UA_StatusCode
__UA_Server_write(UA_Server *server, const UA_NodeId *nodeId,
                  const UA_AttributeId attributeId,
                  const UA_DataType *attr_type, const void *attr);

static UA_INLINE UA_StatusCode
UA_Server_writeBrowseName(UA_Server *server, const UA_NodeId nodeId,
                          const UA_QualifiedName browseName) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_BROWSENAME,
                             &UA_TYPES[UA_TYPES_QUALIFIEDNAME], &browseName);
}

static UA_INLINE UA_StatusCode
UA_Server_writeDisplayName(UA_Server *server, const UA_NodeId nodeId,
                           const UA_LocalizedText displayName) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_DISPLAYNAME,
                             &UA_TYPES[UA_TYPES_LOCALIZEDTEXT], &displayName);
}

static UA_INLINE UA_StatusCode
UA_Server_writeDescription(UA_Server *server, const UA_NodeId nodeId,
                           const UA_LocalizedText description) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_DESCRIPTION,
                             &UA_TYPES[UA_TYPES_LOCALIZEDTEXT], &description);
}

static UA_INLINE UA_StatusCode
UA_Server_writeWriteMask(UA_Server *server, const UA_NodeId nodeId,
                         const UA_UInt32 writeMask) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_WRITEMASK,
                             &UA_TYPES[UA_TYPES_UINT32], &writeMask);
}

static UA_INLINE UA_StatusCode
UA_Server_writeIsAbstract(UA_Server *server, const UA_NodeId nodeId,
                          const UA_Boolean isAbstract) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_ISABSTRACT,
                             &UA_TYPES[UA_TYPES_BOOLEAN], &isAbstract);
}

static UA_INLINE UA_StatusCode
UA_Server_writeInverseName(UA_Server *server, const UA_NodeId nodeId,
                           const UA_LocalizedText inverseName) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_INVERSENAME,
                             &UA_TYPES[UA_TYPES_LOCALIZEDTEXT], &inverseName);
}

static UA_INLINE UA_StatusCode
UA_Server_writeEventNotifier(UA_Server *server, const UA_NodeId nodeId,
                             const UA_Byte eventNotifier) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_EVENTNOTIFIER,
                             &UA_TYPES[UA_TYPES_BYTE], &eventNotifier);
}

static UA_INLINE UA_StatusCode
UA_Server_writeValue(UA_Server *server, const UA_NodeId nodeId,
                     const UA_Variant value) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_VALUE,
```

```
                           &UA_TYPES[UA_TYPES_VARIANT], &value);
}

static UA_INLINE UA_StatusCode
UA_Server_writeDataType(UA_Server *server, const UA_NodeId nodeId,
                        const UA_NodeId dataType) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_DATATYPE,
                             &UA_TYPES[UA_TYPES_NODEID], &dataType);
}

static UA_INLINE UA_StatusCode
UA_Server_writeValueRank(UA_Server *server, const UA_NodeId nodeId,
                         const UA_Int32 valueRank) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_VALUERANK,
                             &UA_TYPES[UA_TYPES_INT32], &valueRank);
}

static UA_INLINE UA_StatusCode
UA_Server_writeArrayDimensions(UA_Server *server, const UA_NodeId nodeId,
                               const UA_Variant arrayDimensions) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_VALUE,
                             &UA_TYPES[UA_TYPES_VARIANT], &arrayDimensions);
}

static UA_INLINE UA_StatusCode
UA_Server_writeAccessLevel(UA_Server *server, const UA_NodeId nodeId,
                           const UA_Byte accessLevel) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_ACCESSLEVEL,
                             &UA_TYPES[UA_TYPES_BYTE], &accessLevel);
}

static UA_INLINE UA_StatusCode
UA_Server_writeMinimumSamplingInterval(UA_Server *server, const UA_NodeId nodeId,
                                       const UA_Double miniumSamplingInterval) {
    return __UA_Server_write(server, &nodeId,
                             UA_ATTRIBUTEID_MINIMUMSAMPLINGINTERVAL,
                             &UA_TYPES[UA_TYPES_DOUBLE],
                             &miniumSamplingInterval);
}

static UA_INLINE UA_StatusCode
UA_Server_writeExecutable(UA_Server *server, const UA_NodeId nodeId,
                          const UA_Boolean executable) {
    return __UA_Server_write(server, &nodeId, UA_ATTRIBUTEID_EXECUTABLE,
                             &UA_TYPES[UA_TYPES_BOOLEAN], &executable); }
```

8.5 Browsing

```
UA_BrowseResult
UA_Server_browse(UA_Server *server, UA_UInt32 maxrefs,
                 const UA_BrowseDescription *descr);

UA_BrowseResult
UA_Server_browseNext(UA_Server *server, UA_Boolean releaseContinuationPoint,
                     const UA_ByteString *continuationPoint);

UA_BrowsePathResult
UA_Server_translateBrowsePathToNodeIds(UA_Server *server,
```

```
                                    const UA_BrowsePath *browsePath);

#ifndef HAVE_NODEITER_CALLBACK
#define HAVE_NODEITER_CALLBACK
/* Iterate over all nodes referenced by parentNodeId by calling the callback
 * function for each child node (in ifdef because GCC/CLANG handle include order
 * differently) */
typedef UA_StatusCode
(*UA_NodeIteratorCallback)(UA_NodeId childId, UA_Boolean isInverse,
                           UA_NodeId referenceTypeId, void *handle);
#endif

UA_StatusCode
UA_Server_forEachChildNodeCall(UA_Server *server, UA_NodeId parentNodeId,
                               UA_NodeIteratorCallback callback, void *handle);


#ifdef UA_ENABLE_DISCOVERY
```

8.6 Discovery

```
/* Register the given server instance at the discovery server.
 * This should be called periodically.
 * The semaphoreFilePath is optional. If the given file is deleted,
 * the server will automatically be unregistered. This could be
 * for example a pid file which is deleted if the server crashes.
 *
 * When the server shuts down you need to call unregister.
 *
 * @param server
 * @param discoveryServerUrl if set to NULL, the default value
 *        'opc.tcp://localhost:4840' will be used
 * @param semaphoreFilePath optional parameter pointing to semaphore file. */
UA_StatusCode
UA_Server_register_discovery(UA_Server *server, const char* discoveryServerUrl,
                             const char* semaphoreFilePath);

/* Unregister the given server instance from the discovery server.
 * This should only be called when the server is shutting down.
 * @param server
 * @param discoveryServerUrl if set to NULL, the default value
 *        'opc.tcp://localhost:4840' will be used */
UA_StatusCode
UA_Server_unregister_discovery(UA_Server *server, const char* discoveryServerUrl);

 /* Adds a periodic callback to register the server with the LDS (local discovery
→server)
  * periodically. The interval between each register call is given as second
→parameter.
  * It should be 10 minutes by default (= 10*60*1000).
  *
  * The delayFirstRegisterMs parameter indicates the delay for the first register
→call.
  * If it is 0, the first register call will be after intervalMs milliseconds,
  * otherwise the server's first register will be after delayFirstRegisterMs.
  *
  * When you manually unregister the server, you also need to cancel the
  * periodic callback, otherwise it will be automatically be registered again.
  *
```

```
 * If you call this method multiple times for the same discoveryServerUrl, the
→older
 * periodic callback will be removed.
 *
 * @param server
 * @param discoveryServerUrl if set to NULL, the default value
 *         'opc.tcp://localhost:4840' will be used
 * @param intervalMs
 * @param delayFirstRegisterMs
 * @param periodicCallbackId */
UA_StatusCode
UA_Server_addPeriodicServerRegisterCallback(UA_Server *server, const char*
→discoveryServerUrl,
                                            UA_UInt32 intervalMs,
                                            UA_UInt32 delayFirstRegisterMs,
                                            UA_UInt64 *periodicCallbackId);

/* Callback for RegisterServer. Data is passed from the register call */
typedef void (*UA_Server_registerServerCallback)(const UA_RegisteredServer
→*registeredServer,
                                                  void* data);

/* Set the callback which is called if another server registeres or unregisters
 * with this instance. If called multiple times, previous data will be
 * overwritten.
 *
 * @param server
 * @param cb the callback
 * @param data data passed to the callback
 * @return UA_STATUSCODE_SUCCESS on success */
void
UA_Server_setRegisterServerCallback(UA_Server *server, UA_Server_
→registerServerCallback cb,
                                    void* data);

#ifdef UA_ENABLE_DISCOVERY_MULTICAST

/* Callback for server detected through mDNS. Data is passed from the register
 * call
 *
 * @param isServerAnnounce indicates if the server has just been detected. If
 *        set to false, this means the server is shutting down.
 * @param isTxtReceived indicates if we already received the corresponding TXT
 *        record with the path and caps data */
typedef void (*UA_Server_serverOnNetworkCallback)(const UA_ServerOnNetwork
→*serverOnNetwork,
                                                   UA_Boolean isServerAnnounce,
                                                   UA_Boolean isTxtReceived, void*
→data);

/* Set the callback which is called if another server is found through mDNS or
 * deleted. It will be called for any mDNS message from the remote server, thus
 * it may be called multiple times for the same instance. Also the SRV and TXT
 * records may arrive later, therefore for the first call the server
 * capabilities may not be set yet. If called multiple times, previous data will
 * be overwritten.
 *
 * @param server
 * @param cb the callback
 * @param data data passed to the callback
 * @return UA_STATUSCODE_SUCCESS on success */
```

```
void
UA_Server_setServerOnNetworkCallback(UA_Server *server,
                                     UA_Server_serverOnNetworkCallback cb,
                                     void* data);

#endif /* UA_ENABLE_DISCOVERY_MULTICAST */

#endif /* UA_ENABLE_DISCOVERY */
```

8.7 Information Model Callbacks

There are three places where a callback from an information model to user-defined code can happen.

- Custom node constructors and destructors
- Linking VariableNodes with an external data source
- MethodNode callbacks

8.7.1 Node Lifecycle: Constructors, Destructors and Node Contexts

To finalize the instantiation of a node, a (user-defined) constructor callback is executed. There can be both a global constructor for all nodes and node-type constructor specific to the TypeDefinition of the new node (attached to an ObjectTypeNode or VariableTypeNode).

In the hierarchy of ObjectTypes and VariableTypes, only the constructor of the (lowest) type defined for the new node is executed. Note that every Object and Variable can have only one `isTypeOf` reference. But type-nodes can technically have several `hasSubType` references to implement multiple inheritance. Issues of (multiple) inheritance in the constructor need to be solved by the user.

When a node is destroyed, the node-type destructor is called before the global destructor. So the overall node lifecycle is as follows:

1. Global Constructor (set in the server config)
2. Node-Type Constructor (for VariableType or ObjectTypes)
3. (Usage-period of the Node)
4. Node-Type Destructor
5. Global Destructor

The constructor and destructor callbacks can be set to `NULL` and are not used in that case. If the node-type constructor fails, the global destructor will be called before removing the node. The destructors are assumed to never fail.

Every node carries a user-context and a constructor-context pointer. The user-context is used to attach custom data to a node. But the (user-defined) constructors and destructors may replace the user-context pointer if they wish to do so. The initial value for the constructor-context is `NULL`. When the `AddNodes` service is used over the network, the user-context pointer of the new node is also initially set to `NULL`.

```
/* To be set in the server config. */
typedef struct {
    /* Can be NULL. May replace the nodeContext */
    UA_StatusCode (*constructor)(UA_Server *server,
                                 const UA_NodeId *sessionId, void *sessionContext,
                                 const UA_NodeId *nodeId, void **nodeContext);

    /* Can be NULL. The context cannot be replaced since the node is destroyed
     * immediately afterwards anyway. */
```

```c
    void (*destructor)(UA_Server *server,
                       const UA_NodeId *sessionId, void *sessionContext,
                       const UA_NodeId *nodeId, void *nodeContext);
} UA_GlobalNodeLifecycle;

typedef struct {
    /* Can be NULL. May replace the nodeContext */
    UA_StatusCode (*constructor)(UA_Server *server,
                                 const UA_NodeId *sessionId, void *sessionContext,
                                 const UA_NodeId *typeNodeId, void␣
→*typeNodeContext,
                                 const UA_NodeId *nodeId, void **nodeContext);

    /* Can be NULL. May replace the nodeContext. */
    void (*destructor)(UA_Server *server,
                       const UA_NodeId *sessionId, void *sessionContext,
                       const UA_NodeId *typeNodeId, void *typeNodeContext,
                       const UA_NodeId *nodeId, void **nodeContext);
} UA_NodeTypeLifecycle;

UA_StatusCode
UA_Server_setNodeTypeLifecycle(UA_Server *server, UA_NodeId nodeId,
                               UA_NodeTypeLifecycle lifecycle);

UA_StatusCode
UA_Server_getNodeContext(UA_Server *server, UA_NodeId nodeId,
                         void **nodeContext);

/* Careful! The user has to ensure that the destructor callbacks still work. */
UA_StatusCode
UA_Server_setNodeContext(UA_Server *server, UA_NodeId nodeId,
                         void *nodeContext);
```

8.7.2 Data Source Callback

The server has a unique way of dealing with the content of variables. Instead of storing a variant attached to the variable node, the node can point to a function with a local data provider. Whenever the value attribute is read, the function will be called and asked to provide a UA_DataValue return value that contains the value content and additional timestamps.

It is expected that the read callback is implemented. The write callback can be set to a null-pointer.

```c
typedef struct {
    /* Copies the data from the source into the provided value.
     *
     * !! ZERO-COPY OPERATIONS POSSIBLE !!
     * It is not required to return a copy of the actual content data. You can
     * return a pointer to memory owned by the user. Memory can be reused
     * between read callbacks of a DataSource, as the result is already encoded
     * on the network buffer between each read operation.
     *
     * To use zero-copy reads, set the value of the `value->value` Variant
     * without copying, e.g. with `UA_Variant_setScalar`. Then, also set
     * `value->value.storageType` to `UA_VARIANT_DATA_NODELETE` to prevent the
     * memory being cleaned up. Don't forget to also set `value->hasValue` to
     * true to indicate the presence of a value.
     *
     * @param handle An optional pointer to user-defined data for the
     *        specific data source
     * @param nodeid Id of the read node
```

```
    * @param includeSourceTimeStamp If true, then the datasource is expected to
    *        set the source timestamp in the returned value
    * @param range If not null, then the datasource shall return only a
    *        selection of the (nonscalar) data. Set
    *        UA_STATUSCODE_BADINDEXRANGEINVALID in the value if this does not
    *        apply.
    * @param value The (non-null) DataValue that is returned to the client. The
    *        data source sets the read data, the result status and optionally a
    *        sourcetimestamp.
    * @return Returns a status code for logging. Error codes intended for the
    *         original caller are set in the value. If an error is returned,
    *         then no releasing of the value is done. */
   UA_StatusCode (*read)(UA_Server *server, const UA_NodeId *sessionId,
                         void *sessionContext, const UA_NodeId *nodeId,
                         void *nodeContext, UA_Boolean includeSourceTimeStamp,
                         const UA_NumericRange *range, UA_DataValue *value);

   /* Write into a data source. This method pointer can be NULL if the
    * operation is unsupported.
    *
    * @param handle An optional pointer to user-defined data for the
    *        specific data source
    * @param nodeid Id of the node being written to
    * @param data The data to be written into the data source
    * @param range An optional data range. If the data source is scalar or does
    *        not support writing of ranges, then an error code is returned.
    * @return Returns a status code that is returned to the user */
   UA_StatusCode (*write)(UA_Server *server, const UA_NodeId *sessionId,
                          void *sessionContext, const UA_NodeId *nodeId,
                          void *nodeContext, const UA_NumericRange *range,
                          const UA_DataValue *value);
} UA_DataSource;

UA_StatusCode
UA_Server_setVariableNode_dataSource(UA_Server *server, const UA_NodeId nodeId,
                                     const UA_DataSource dataSource);
```

8.7.3 Value Callback

Value Callbacks can be attached to variable and variable type nodes. If not NULL, they are called before reading and after writing respectively.

```
typedef struct {
    /* Called before the value attribute is read. It is possible to write into the
     * value attribute during onRead (using the write service). The node is
     * re-opened afterwards so that changes are considered in the following read
     * operation.
     *
     * @param handle Points to user-provided data for the callback.
     * @param nodeid The identifier of the node.
     * @param data Points to the current node value.
     * @param range Points to the numeric range the client wants to read from
     *        (or NULL). */
    void (*onRead)(UA_Server *server, const UA_NodeId *sessionId,
                   void *sessionContext, const UA_NodeId *nodeid,
                   void *nodeContext, const UA_NumericRange *range,
                   const UA_DataValue *value);

    /* Called after writing the value attribute. The node is re-opened after
```

```
    * writing so that the new value is visible in the callback.
    *
    * @param server The server executing the callback
    * @sessionId The identifier of the session
    * @sessionContext Additional data attached to the session
    *                 in the access control layer
    * @param nodeid The identifier of the node.
    * @param nodeUserContext Additional data attached to the node by
    *        the user.
    * @param nodeConstructorContext Additional data attached to the node
    *        by the type constructor(s).
    * @param range Points to the numeric range the client wants to write to (or
    *        NULL). */
    void (*onWrite)(UA_Server *server, const UA_NodeId *sessionId,
                    void *sessionContext, const UA_NodeId *nodeId,
                    void *nodeContext, const UA_NumericRange *range,
                    const UA_DataValue *data);
} UA_ValueCallback;

UA_StatusCode
UA_Server_setVariableNode_valueCallback(UA_Server *server,
                                        const UA_NodeId nodeId,
                                        const UA_ValueCallback callback);
```

8.7.4 Method Callbacks

Method callbacks are set to *NULL* (not executable) when a method node is added over the network. In theory, it is possible to add a callback via UA_Server_setMethodNode_callback within the global constructor when adding methods over the network is really wanted.

```
typedef UA_StatusCode
(*UA_MethodCallback)(UA_Server *server, const UA_NodeId *sessionId,
                     void *sessionContext, const UA_NodeId *methodId,
                     void *methodContext, const UA_NodeId *objectId,
                     void *objectContext, size_t inputSize,
                     const UA_Variant *input, size_t outputSize,
                     UA_Variant *output);

#ifdef UA_ENABLE_METHODCALLS

UA_StatusCode
UA_Server_setMethodNode_callback(UA_Server *server,
                                 const UA_NodeId methodNodeId,
                                 UA_MethodCallback methodCallback);
UA_CallMethodResult
UA_Server_call(UA_Server *server, const UA_CallMethodRequest *request);

#endif
```

8.8 Node Addition and Deletion

When creating dynamic node instances at runtime, chances are that you will not care about the specific NodeId of the new node, as long as you can reference it later. When passing numeric NodeIds with a numeric identifier 0, the stack evaluates this as "select a random unassigned numeric NodeId in that namespace". To find out which NodeId was actually assigned to the new node, you may pass a pointer *outNewNodeId*, which will (after a successful node insertion) contain the nodeId of the new node. You may also pass a NULL pointer if this result is not needed.

See the Section *Node Lifecycle: Constructors, Destructors and Node Contexts* on constructors and on attaching user-defined data to nodes.

The methods for node addition and deletion take mostly const arguments that are not modified. When creating a node, a deep copy of the node identifier, node attributes, etc. is created. Therefore, it is possible to call for example `UA_Server_addVariablenode` with a value attribute (a *Variant*) pointing to a memory location on the stack. If you need changes to a variable value to manifest at a specific memory location, please use a *Data Source Callback* or a *Value Callback*.

```c
/* Protect against redundant definitions for server/client */
#ifndef UA_DEFAULT_ATTRIBUTES_DEFINED
#define UA_DEFAULT_ATTRIBUTES_DEFINED
/* The default for variables is "BaseDataType" for the datatype, -2 for the
 * valuerank and a read-accesslevel. */
extern const UA_VariableAttributes UA_VariableAttributes_default;
extern const UA_VariableTypeAttributes UA_VariableTypeAttributes_default;
/* Methods are executable by default */
extern const UA_MethodAttributes UA_MethodAttributes_default;
/* The remaining attribute definitions are currently all zeroed out */
extern const UA_ObjectAttributes UA_ObjectAttributes_default;
extern const UA_ObjectTypeAttributes UA_ObjectTypeAttributes_default;
extern const UA_ReferenceTypeAttributes UA_ReferenceTypeAttributes_default;
extern const UA_DataTypeAttributes UA_DataTypeAttributes_default;
extern const UA_ViewAttributes UA_ViewAttributes_default;
#endif

/* Don't use this function. There are typed versions as inline functions. */
UA_StatusCode
__UA_Server_addNode(UA_Server *server, const UA_NodeClass nodeClass,
                    const UA_NodeId *requestedNewNodeId,
                    const UA_NodeId *parentNodeId,
                    const UA_NodeId *referenceTypeId,
                    const UA_QualifiedName browseName,
                    const UA_NodeId *typeDefinition,
                    const UA_NodeAttributes *attr,
                    const UA_DataType *attributeType,
                    void *nodeContext, UA_NodeId *outNewNodeId);

static UA_INLINE UA_StatusCode
UA_Server_addVariableNode(UA_Server *server, const UA_NodeId requestedNewNodeId,
                          const UA_NodeId parentNodeId,
                          const UA_NodeId referenceTypeId,
                          const UA_QualifiedName browseName,
                          const UA_NodeId typeDefinition,
                          const UA_VariableAttributes attr,
                          void *nodeContext, UA_NodeId *outNewNodeId) {
    return __UA_Server_addNode(server, UA_NODECLASS_VARIABLE, &requestedNewNodeId,
                               &parentNodeId, &referenceTypeId, browseName,
                               &typeDefinition, (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_VARIABLEATTRIBUTES],
                               nodeContext, outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Server_addVariableTypeNode(UA_Server *server,
                              const UA_NodeId requestedNewNodeId,
                              const UA_NodeId parentNodeId,
                              const UA_NodeId referenceTypeId,
                              const UA_QualifiedName browseName,
                              const UA_NodeId typeDefinition,
                              const UA_VariableTypeAttributes attr,
                              void *nodeContext, UA_NodeId *outNewNodeId) {
```

(continues on next page)

(continued from previous page)

```c
    return __UA_Server_addNode(server, UA_NODECLASS_VARIABLETYPE,
                               &requestedNewNodeId, &parentNodeId, &
↪referenceTypeId,
                               browseName, &typeDefinition,
                               (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_VARIABLETYPEATTRIBUTES],
                               nodeContext, outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Server_addObjectNode(UA_Server *server, const UA_NodeId requestedNewNodeId,
                        const UA_NodeId parentNodeId,
                        const UA_NodeId referenceTypeId,
                        const UA_QualifiedName browseName,
                        const UA_NodeId typeDefinition,
                        const UA_ObjectAttributes attr,
                        void *nodeContext, UA_NodeId *outNewNodeId) {
    return __UA_Server_addNode(server, UA_NODECLASS_OBJECT, &requestedNewNodeId,
                               &parentNodeId, &referenceTypeId, browseName,
                               &typeDefinition, (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_OBJECTATTRIBUTES],
                               nodeContext, outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Server_addObjectTypeNode(UA_Server *server, const UA_NodeId requestedNewNodeId,
                            const UA_NodeId parentNodeId,
                            const UA_NodeId referenceTypeId,
                            const UA_QualifiedName browseName,
                            const UA_ObjectTypeAttributes attr,
                            void *nodeContext, UA_NodeId *outNewNodeId) {
    return __UA_Server_addNode(server, UA_NODECLASS_OBJECTTYPE, &
↪requestedNewNodeId,
                               &parentNodeId, &referenceTypeId, browseName,
                               &UA_NODEID_NULL, (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_OBJECTTYPEATTRIBUTES],
                               nodeContext, outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Server_addViewNode(UA_Server *server, const UA_NodeId requestedNewNodeId,
                      const UA_NodeId parentNodeId,
                      const UA_NodeId referenceTypeId,
                      const UA_QualifiedName browseName,
                      const UA_ViewAttributes attr,
                      void *nodeContext, UA_NodeId *outNewNodeId) {
    return __UA_Server_addNode(server, UA_NODECLASS_VIEW, &requestedNewNodeId,
                               &parentNodeId, &referenceTypeId, browseName,
                               &UA_NODEID_NULL, (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_VIEWATTRIBUTES],
                               nodeContext, outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Server_addReferenceTypeNode(UA_Server *server,
                               const UA_NodeId requestedNewNodeId,
                               const UA_NodeId parentNodeId,
                               const UA_NodeId referenceTypeId,
                               const UA_QualifiedName browseName,
                               const UA_ReferenceTypeAttributes attr,
                               void *nodeContext, UA_NodeId *outNewNodeId) {
```

(continues on next page)

```
    return __UA_Server_addNode(server, UA_NODECLASS_REFERENCETYPE,
                              &requestedNewNodeId, &parentNodeId, &
→referenceTypeId,
                              browseName, &UA_NODEID_NULL,
                              (const UA_NodeAttributes*)&attr,
                              &UA_TYPES[UA_TYPES_REFERENCETYPEATTRIBUTES],
                              nodeContext, outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Server_addDataTypeNode(UA_Server *server,
                          const UA_NodeId requestedNewNodeId,
                          const UA_NodeId parentNodeId,
                          const UA_NodeId referenceTypeId,
                          const UA_QualifiedName browseName,
                          const UA_DataTypeAttributes attr,
                          void *nodeContext, UA_NodeId *outNewNodeId) {
    return __UA_Server_addNode(server, UA_NODECLASS_DATATYPE, &requestedNewNodeId,
                              &parentNodeId, &referenceTypeId, browseName,
                              &UA_NODEID_NULL, (const UA_NodeAttributes*)&attr,
                              &UA_TYPES[UA_TYPES_DATATYPEATTRIBUTES],
                              nodeContext, outNewNodeId);
}

UA_StatusCode
UA_Server_addDataSourceVariableNode(UA_Server *server,
                                    const UA_NodeId requestedNewNodeId,
                                    const UA_NodeId parentNodeId,
                                    const UA_NodeId referenceTypeId,
                                    const UA_QualifiedName browseName,
                                    const UA_NodeId typeDefinition,
                                    const UA_VariableAttributes attr,
                                    const UA_DataSource dataSource,
                                    void *nodeContext, UA_NodeId *outNewNodeId);

UA_StatusCode
UA_Server_addMethodNodeEx(UA_Server *server, const UA_NodeId requestedNewNodeId,
                          const UA_NodeId parentNodeId,
                          const UA_NodeId referenceTypeId,
                          const UA_QualifiedName browseName,
                          const UA_MethodAttributes attr, UA_MethodCallback method,
                          size_t inputArgumentsSize, const UA_Argument␣
→*inputArguments,
                          const UA_NodeId inputArgumentsRequestedNewNodeId,
                          UA_NodeId *inputArgumentsOutNewNodeId,
                          size_t outputArgumentsSize, const UA_Argument␣
→*outputArguments,
                          const UA_NodeId outputArgumentsRequestedNewNodeId,
                          UA_NodeId *outputArgumentsOutNewNodeId,
                          void *nodeContext, UA_NodeId *outNewNodeId);

static UA_INLINE UA_StatusCode
UA_Server_addMethodNode(UA_Server *server, const UA_NodeId requestedNewNodeId,
                        const UA_NodeId parentNodeId, const UA_NodeId␣
→referenceTypeId,
                        const UA_QualifiedName browseName, const UA_
→MethodAttributes attr,
                        UA_MethodCallback method,
                        size_t inputArgumentsSize, const UA_Argument␣
→*inputArguments,
                        size_t outputArgumentsSize, const UA_Argument␣
→*outputArguments,
```

```
                            void *nodeContext, UA_NodeId *outNewNodeId) {
    return UA_Server_addMethodNodeEx(server, requestedNewNodeId, parentNodeId,
                                     referenceTypeId, browseName, attr, method,
                                     inputArgumentsSize, inputArguments, UA_NODEID_
↪NULL, NULL,
                                     outputArgumentsSize, outputArguments, UA_
↪NODEID_NULL, NULL,
                                     nodeContext, outNewNodeId);
}
```

The method pair UA_Server_addNode_begin and _finish splits the AddNodes service in two parts. This is useful if the node shall be modified before finish the instantiation. For example to add children with specific NodeIds. Otherwise, mandatory children (e.g. of an ObjectType) are added with pseudo-random unique NodeIds. Existing children are detected during the _finish part via their matching BrowseName.

The _begin method:

- prepares the node and adds it to the nodestore

- copies some unassigned attributes from the TypeDefinition node internally

- adds the references to the parent (and the TypeDefinition if applicable)

- performs type-checking of variables.

You can add an object node without a parent if you set the parentNodeId and referenceTypeId to UA_NODE_ID_NULL. Then you need to add the parent reference and hasTypeDef reference yourself before calling the _finish method. Not that this is only allowed for object nodes.

The _finish method:

- copies mandatory children

- calls the node constructor(s) at the end

- may remove the node if it encounters an error.

The special UA_Server_addMethodNode_finish method needs to be used for method nodes, since there you need to explicitly specifiy the input and output arguments which are added in the finish step (if not yet already there)

`VariableAttributes` for variables, `ObjectAttributes` for objects, and so on. Missing attributes are taken from the TypeDefinition node if applicable.

```
UA_StatusCode
UA_Server_addNode_begin(UA_Server *server, const UA_NodeClass nodeClass,
                        const UA_NodeId requestedNewNodeId,
                        const UA_NodeId parentNodeId,
                        const UA_NodeId referenceTypeId,
                        const UA_QualifiedName browseName,
                        const UA_NodeId typeDefinition,
                        const void *attr, const UA_DataType *attributeType,
                        void *nodeContext, UA_NodeId *outNewNodeId);

UA_StatusCode
UA_Server_addNode_finish(UA_Server *server, const UA_NodeId nodeId);

UA_StatusCode
UA_Server_addMethodNode_finish(UA_Server *server, const UA_NodeId nodeId,
                        UA_MethodCallback method,
                        size_t inputArgumentsSize, const UA_Argument*_
↪inputArguments,
                        size_t outputArgumentsSize, const UA_Argument*_
↪outputArguments);

/* Deletes a node and optionally all references leading to the node. */
```

```
UA_StatusCode
UA_Server_deleteNode(UA_Server *server, const UA_NodeId nodeId,
                     UA_Boolean deleteReferences);
```

8.9 Reference Management

```
UA_StatusCode
UA_Server_addReference(UA_Server *server, const UA_NodeId sourceId,
                       const UA_NodeId refTypeId,
                       const UA_ExpandedNodeId targetId, UA_Boolean isForward);

UA_StatusCode
UA_Server_deleteReference(UA_Server *server, const UA_NodeId sourceNodeId,
                          const UA_NodeId referenceTypeId, UA_Boolean isForward,
                          const UA_ExpandedNodeId targetNodeId,
                          UA_Boolean deleteBidirectional);
```

8.10 Utility Functions

```
/* Add a new namespace to the server. Returns the index of the new namespace */
UA_UInt16 UA_Server_addNamespace(UA_Server *server, const char* name);
```

8.11 Deprecated Server API

This file contains outdated API definitions that are kept for backwards compatibility. Please switch to the new API, as the following definitions will be removed eventually.

8.11.1 UA_Job API

UA_Job was replaced since it unnecessarily exposed server internals to the end-user. Please use plain UA_ServerCallbacks instead. The following UA_Job definition contains just the fraction of the original struct that was useful to end-users.

```
typedef enum {
    UA_JOBTYPE_METHODCALL
} UA_JobType;

typedef struct {
    UA_JobType type;
    union {
        struct {
            void *data;
            UA_ServerCallback method;
        } methodCall;
    } job;
} UA_Job;

UA_DEPRECATED static UA_INLINE UA_StatusCode
UA_Server_addRepeatedJob(UA_Server *server, UA_Job job,
                         UA_UInt32 interval, UA_Guid *jobId) {
    return UA_Server_addRepeatedCallback(server, job.job.methodCall.method,
```

```
                                              job.job.methodCall.data, interval,
                                              (UA_UInt64*)(uintptr_t)jobId);
}


UA_DEPRECATED static UA_INLINE UA_StatusCode
UA_Server_removeRepeatedJob(UA_Server *server, UA_Guid jobId) {
    return UA_Server_removeRepeatedCallback(server,
                                            *(UA_UInt64*)(uintptr_t)&jobId);
}
```

Client

The client implementation allows remote access to all OPC UA services. For convenience, some functionality has been wrapped in *high-level abstractions*.

However: At this time, the client does not yet contain its own thread or event-driven main-loop. So the client will not perform any actions automatically in the background. This is especially relevant for subscriptions. The user will have to periodically call *UA_Client_Subscriptions_manuallySendPublishRequest*. See also *here*.

9.1 Client Configuration

The client configuration is used for setting connection parameters and additional settings used by the client. The configuration should not be modified after it is passed to a client. Currently, only one client can use a configuration at a time.

Examples for configurations are provided in the `/plugins` folder. The usual usage is as follows:

1. Create a client configuration with default settings as a starting point

2. Modifiy the configuration, e.g. modifying the timeout

3. Instantiate a client with it

4. After shutdown of the client, clean up the configuration (free memory)

The *Tutorials* provide a good starting point for this.

```c
typedef enum {
    UA_CLIENTSTATE_DISCONNECTED,          /* The client is disconnected */
    UA_CLIENTSTATE_CONNECTED,             /* A TCP connection to the server is open␣
↪*/
    UA_CLIENTSTATE_SECURECHANNEL,         /* A SecureChannel to the server is open */
    UA_CLIENTSTATE_SESSION,               /* A session with the server is open */
    UA_CLIENTSTATE_SESSION_RENEWED        /* A session with the server is open␣
↪(renewed) */
} UA_ClientState;



struct UA_Client;
typedef struct UA_Client UA_Client;
```

9.1.1 Client Lifecycle callback

```
typedef void (*UA_ClientStateCallback)(UA_Client *client, UA_ClientState
→clientState);
```

9.1.2 Subscription Inactivity callback

```
#ifdef UA_ENABLE_SUBSCRIPTIONS
typedef void (*UA_SubscriptionInactivityCallback)(UA_Client *client, UA_UInt32
→subscriptionId, void *subContext);
#endif
```

9.1.3 Inactivity callback

```
typedef void (*UA_InactivityCallback)(UA_Client *client);
```

9.1.4 Client Configuration Data

```
typedef struct UA_ClientConfig {
    UA_UInt32 timeout;               /* ASync + Sync response timeout in ms */
    UA_UInt32 secureChannelLifeTime; /* Lifetime in ms (then the channel needs
                                        to be renewed) */
    UA_Logger logger;
    UA_ConnectionConfig localConnectionConfig;
    UA_ConnectClientConnection connectionFunc;

    /* Custom DataTypes */
    size_t customDataTypesSize;
    const UA_DataType *customDataTypes;

    /* Callback function */
    UA_ClientStateCallback stateCallback;
#ifdef UA_ENABLE_SUBSCRIPTIONS
```

When outStandingPublishRequests is greater than 0, the server automatically create publishRequest when UA_Client_runAsync is called. If the client don't receive a publishResponse after :

(sub->publishingInterval * sub->maxKeepAliveCount) + client->config.timeout)

then, the client call subscriptionInactivityCallback The connection can be closed, this in an attempt to recreate a healthy connection.

```
    UA_SubscriptionInactivityCallback subscriptionInactivityCallback;
#endif
```

When connectivityCheckInterval is greater than 0, every connectivityCheckInterval (in ms), a async read request is performed on the server. inactivityCallback is called when the client receive no response for this read request The connection can be closed, this in an attempt to recreate a healthy connection.

```
    UA_InactivityCallback inactivityCallback;

    void *clientContext;

#ifdef UA_ENABLE_SUBSCRIPTIONS
    /* number of PublishResponse standing in the sever */
    /* 0 = background task disabled                    */
```

```
    UA_UInt16 outStandingPublishRequests;
#endif
```

connectivity check interval in ms 0 = background task disabled

```
    UA_UInt32 connectivityCheckInterval;
} UA_ClientConfig;


/* Get the client configuration from the configuration plugin. Used by the
 * server when it needs client functionality to register to a discovery server
 * or when the server needs to create a client for other purposes
 *
 * @return The client configuration structure */
UA_ClientConfig
UA_Server_getClientConfig(void);
```

9.2 Client Lifecycle

```
/* Create a new client */
UA_Client *
UA_Client_new(UA_ClientConfig config);

/* Get the client connection status */
UA_ClientState
UA_Client_getState(UA_Client *client);

/* Get the client context */
void *
UA_Client_getContext(UA_Client *client);

/* Reset a client */
void
UA_Client_reset(UA_Client *client);

/* Delete a client */
void
UA_Client_delete(UA_Client *client);
```

9.3 Connect to a Server

```
/* Connect to the server
 *
 * @param client to use
 * @param endpointURL to connect (for example "opc.tcp://localhost:4840")
 * @return Indicates whether the operation succeeded or returns an error code */
UA_StatusCode
UA_Client_connect(UA_Client *client, const char *endpointUrl);

/* Connect to the selected server with the given username and password
 *
 * @param client to use
 * @param endpointURL to connect (for example "opc.tcp://localhost:4840")
 * @param username
 * @param password
```

```
 * @return Indicates whether the operation succeeded or returns an error code */
UA_StatusCode
UA_Client_connect_username(UA_Client *client, const char *endpointUrl,
                           const char *username, const char *password);

/* Disconnect and close a connection to the selected server */
UA_StatusCode
UA_Client_disconnect(UA_Client *client);

/* Close a connection to the selected server */
UA_StatusCode
UA_Client_close(UA_Client *client);

/* Renew the underlying secure channel */
UA_StatusCode
UA_Client_manuallyRenewSecureChannel(UA_Client *client);
```

9.4 Discovery

```
/* Gets a list of endpoints of a server
 *
 * @param client to use. Must be connected to the same endpoint given in
 *        serverUrl or otherwise in disconnected state.
 * @param serverUrl url to connect (for example "opc.tcp://localhost:4840")
 * @param endpointDescriptionsSize size of the array of endpoint descriptions
 * @param endpointDescriptions array of endpoint descriptions that is allocated
 *        by the function (you need to free manually)
 * @return Indicates whether the operation succeeded or returns an error code */
UA_StatusCode
UA_Client_getEndpoints(UA_Client *client, const char *serverUrl,
                       size_t* endpointDescriptionsSize,
                       UA_EndpointDescription** endpointDescriptions);

/* Gets a list of all registered servers at the given server.
 *
 * You can pass an optional filter for serverUris. If the given server is not
→registered,
 * an empty array will be returned. If the server is registered, only that
→application
 * description will be returned.
 *
 * Additionally you can optionally indicate which locale you want for the server
→name
 * in the returned application description. The array indicates the order of
→preference.
 * A server may have localized names.
 *
 * @param client to use. Must be connected to the same endpoint given in
 *        serverUrl or otherwise in disconnected state.
 * @param serverUrl url to connect (for example "opc.tcp://localhost:4840")
 * @param serverUrisSize Optional filter for specific server uris
 * @param serverUris Optional filter for specific server uris
 * @param localeIdsSize Optional indication which locale you prefer
 * @param localeIds Optional indication which locale you prefer
 * @param registeredServersSize size of returned array, i.e., number of found/
→registered servers
 * @param registeredServers array containing found/registered servers
 * @return Indicates whether the operation succeeded or returns an error code */
```

```
UA_StatusCode
UA_Client_findServers(UA_Client *client, const char *serverUrl,
                      size_t serverUrisSize, UA_String *serverUris,
                      size_t localeIdsSize, UA_String *localeIds,
                      size_t *registeredServersSize,
                      UA_ApplicationDescription **registeredServers);


/* Get a list of all known server in the network. Only supported by LDS servers.
 *
 * @param client to use. Must be connected to the same endpoint given in
 * serverUrl or otherwise in disconnected state.
 * @param serverUrl url to connect (for example "opc.tcp://localhost:4840")
 * @param startingRecordId optional. Only return the records with an ID higher
 *        or equal the given. Can be used for pagination to only get a subset of
 *        the full list
 * @param maxRecordsToReturn optional. Only return this number of records

 * @param serverCapabilityFilterSize optional. Filter the returned list to only
 *        get servers with given capabilities, e.g. "LDS"
 * @param serverCapabilityFilter optional. Filter the returned list to only get
 *        servers with given capabilities, e.g. "LDS"
 * @param serverOnNetworkSize size of returned array, i.e., number of
 *        known/registered servers
 * @param serverOnNetwork array containing known/registered servers
 * @return Indicates whether the operation succeeded or returns an error code */
UA_StatusCode
UA_Client_findServersOnNetwork(UA_Client *client, const char *serverUrl,
                               UA_UInt32 startingRecordId, UA_UInt32
→maxRecordsToReturn,
                               size_t serverCapabilityFilterSize, UA_String
→*serverCapabilityFilter,
                               size_t *serverOnNetworkSize, UA_ServerOnNetwork
→**serverOnNetwork);
```

9.5 Services

The raw OPC UA services are exposed to the client. But most of them time, it is better to use the convenience
functions from `ua_client_highlevel.h` that wrap the raw services.

```
/* Don't use this function. Use the type versions below instead. */
void
__UA_Client_Service(UA_Client *client, const void *request,
                    const UA_DataType *requestType, void *response,
                    const UA_DataType *responseType);


/*
 * Attribute Service Set
 * ^^^^^^^^^^^^^^^^^^^^^ */
static UA_INLINE UA_ReadResponse
UA_Client_Service_read(UA_Client *client, const UA_ReadRequest request) {
    UA_ReadResponse response;
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_READREQUEST],
                        &response, &UA_TYPES[UA_TYPES_READRESPONSE]);
    return response;
}


static UA_INLINE UA_WriteResponse
UA_Client_Service_write(UA_Client *client, const UA_WriteRequest request) {
    UA_WriteResponse response;
```

```c
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_WRITEREQUEST],
                        &response, &UA_TYPES[UA_TYPES_WRITERESPONSE]);
    return response;
}

/*
 * Method Service Set
 * ^^^^^^^^^^^^^^^^^^ */
#ifdef UA_ENABLE_METHODCALLS
static UA_INLINE UA_CallResponse
UA_Client_Service_call(UA_Client *client, const UA_CallRequest request) {
    UA_CallResponse response;
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_CALLREQUEST],
                        &response, &UA_TYPES[UA_TYPES_CALLRESPONSE]);
    return response;
}
#endif

/*
 * NodeManagement Service Set
 * ^^^^^^^^^^^^^^^^^^^^^^^^^^ */
static UA_INLINE UA_AddNodesResponse
UA_Client_Service_addNodes(UA_Client *client, const UA_AddNodesRequest request) {
    UA_AddNodesResponse response;
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_ADDNODESREQUEST],
                        &response, &UA_TYPES[UA_TYPES_ADDNODESRESPONSE]);
    return response;
}

static UA_INLINE UA_AddReferencesResponse
UA_Client_Service_addReferences(UA_Client *client,
                                const UA_AddReferencesRequest request) {
    UA_AddReferencesResponse response;
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_ADDREFERENCESREQUEST],
                        &response, &UA_TYPES[UA_TYPES_ADDREFERENCESRESPONSE]);
    return response;
}

static UA_INLINE UA_DeleteNodesResponse
UA_Client_Service_deleteNodes(UA_Client *client,
                              const UA_DeleteNodesRequest request) {
    UA_DeleteNodesResponse response;
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_DELETENODESREQUEST],
                        &response, &UA_TYPES[UA_TYPES_DELETENODESRESPONSE]);
    return response;
}

static UA_INLINE UA_DeleteReferencesResponse
UA_Client_Service_deleteReferences(UA_Client *client,
                                   const UA_DeleteReferencesRequest request) {
    UA_DeleteReferencesResponse response;
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_
DELETEREFERENCESREQUEST],
                        &response, &UA_TYPES[UA_TYPES_DELETEREFERENCESRESPONSE]);
    return response;
}

/*
 * View Service Set
 * ^^^^^^^^^^^^^^^^ */
static UA_INLINE UA_BrowseResponse
```

Chapter 9. Client

```c
UA_Client_Service_browse(UA_Client *client, const UA_BrowseRequest request) {
    UA_BrowseResponse response;
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_BROWSEREQUEST],
                        &response, &UA_TYPES[UA_TYPES_BROWSERESPONSE]);
    return response;
}

static UA_INLINE UA_BrowseNextResponse
UA_Client_Service_browseNext(UA_Client *client,
                             const UA_BrowseNextRequest request) {
    UA_BrowseNextResponse response;
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_BROWSENEXTREQUEST],
                        &response, &UA_TYPES[UA_TYPES_BROWSENEXTRESPONSE]);
    return response;
}

static UA_INLINE UA_TranslateBrowsePathsToNodeIdsResponse
UA_Client_Service_translateBrowsePathsToNodeIds(UA_Client *client,
                        const UA_TranslateBrowsePathsToNodeIdsRequest request) {
    UA_TranslateBrowsePathsToNodeIdsResponse response;
    __UA_Client_Service(client, &request,
                        &UA_TYPES[UA_TYPES_TRANSLATEBROWSEPATHSTONODEIDSREQUEST],
                        &response,
                        &UA_TYPES[UA_TYPES_TRANSLATEBROWSEPATHSTONODEIDSRESPONSE]);
    return response;
}

static UA_INLINE UA_RegisterNodesResponse
UA_Client_Service_registerNodes(UA_Client *client,
                                const UA_RegisterNodesRequest request) {
    UA_RegisterNodesResponse response;
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_REGISTERNODESREQUEST],
                        &response, &UA_TYPES[UA_TYPES_REGISTERNODESRESPONSE]);
    return response;
}

static UA_INLINE UA_UnregisterNodesResponse
UA_Client_Service_unregisterNodes(UA_Client *client,
                                  const UA_UnregisterNodesRequest request) {
    UA_UnregisterNodesResponse response;
    __UA_Client_Service(client, &request,
                        &UA_TYPES[UA_TYPES_UNREGISTERNODESREQUEST],
                        &response, &UA_TYPES[UA_TYPES_UNREGISTERNODESRESPONSE]);
    return response;
}

/*
 * Query Service Set
 * ^^^^^^^^^^^^^^^^^ */
static UA_INLINE UA_QueryFirstResponse
UA_Client_Service_queryFirst(UA_Client *client,
                             const UA_QueryFirstRequest request) {
    UA_QueryFirstResponse response;
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_QUERYFIRSTREQUEST],
                        &response, &UA_TYPES[UA_TYPES_QUERYFIRSTRESPONSE]);
    return response;
}

static UA_INLINE UA_QueryNextResponse
UA_Client_Service_queryNext(UA_Client *client,
                            const UA_QueryNextRequest request) {
```

```
    UA_QueryNextResponse response;
    __UA_Client_Service(client, &request, &UA_TYPES[UA_TYPES_QUERYFIRSTREQUEST],
                        &response, &UA_TYPES[UA_TYPES_QUERYFIRSTRESPONSE]);
    return response;
}
```

9.6 Asynchronous Services

All OPC UA services are asynchronous in nature. So several service calls can be made without waiting for a response first. Responess may come in a different ordering.

```
/* Listen on the network and process arriving asynchronous responses in the
 * background. Internal housekeeping and subscription management is done as
 * well. */
UA_StatusCode
UA_Client_runAsync(UA_Client *client, UA_UInt16 timeout);


typedef void
(*UA_ClientAsyncServiceCallback)(UA_Client *client, void *userdata,
                                 UA_UInt32 requestId, void *response,
                                 const UA_DataType *responseType);


/* Use the type versions of this method. See below. However, the general
 * mechanism of async service calls is explained here.
 *
 * We say that an async service call has been dispatched once this method
 * returns UA_STATUSCODE_GOOD. If there is an error after an async service has
 * been dispatched, the callback is called with an "empty" response where the
 * statusCode has been set accordingly. This is also done if the client is
 * shutting down and the list of dispatched async services is emptied.
 *
 * The statusCode received when the client is shutting down is
 * UA_STATUSCODE_BADSHUTDOWN.
 *
 * The statusCode received when the client don't receive response
 * after specified config->timeout (in ms) is
 * UA_STATUSCODE_BADTIMEOUT.
 *
 * Instead, you can use __UA_Client_AsyncServiceEx to specify
 * a custom timeout
 *
 * The userdata and requestId arguments can be NULL. */
UA_StatusCode
__UA_Client_AsyncService(UA_Client *client, const void *request,
                         const UA_DataType *requestType,
                         UA_ClientAsyncServiceCallback callback,
                         const UA_DataType *responseType,
                         void *userdata, UA_UInt32 *requestId);


/* Use the type versions of this method. See below. However, the general
 * mechanism of async service calls is explained here.
 *
 * We say that an async service call has been dispatched once this method
 * returns UA_STATUSCODE_GOOD. If there is an error after an async service has
 * been dispatched, the callback is called with an "empty" response where the
 * statusCode has been set accordingly. This is also done if the client is
 * shutting down and the list of dispatched async services is emptied.
 *
 * The statusCode received when the client is shutting down is
```

```
 * UA_STATUSCODE_BADSHUTDOWN.
 *
 * The statusCode received when the client don't receive response
 * after specified timeout (in ms) is
 * UA_STATUSCODE_BADTIMEOUT.
 *
 * The timeout can be disabled by setting timeout to 0
 *
 * The userdata and requestId arguments can be NULL. */
UA_StatusCode
__UA_Client_AsyncServiceEx(UA_Client *client, const void *request,
                           const UA_DataType *requestType,
                           UA_ClientAsyncServiceCallback callback,
                           const UA_DataType *responseType,
                           void *userdata, UA_UInt32 *requestId,
                           UA_UInt32 timeout);

static UA_INLINE UA_StatusCode
UA_Client_AsyncService_read(UA_Client *client, const UA_ReadRequest *request,
                            UA_ClientAsyncServiceCallback callback,
                            void *userdata, UA_UInt32 *requestId) {
    return __UA_Client_AsyncService(client, (const void*)request,
                                    &UA_TYPES[UA_TYPES_READREQUEST], callback,
                                    &UA_TYPES[UA_TYPES_READRESPONSE],
                                    userdata, requestId);
}

static UA_INLINE UA_StatusCode
UA_Client_AsyncService_write(UA_Client *client, const UA_WriteRequest *request,
                             UA_ClientAsyncServiceCallback callback,
                             void *userdata, UA_UInt32 *requestId) {
    return __UA_Client_AsyncService(client, (const void*)request,
                                    &UA_TYPES[UA_TYPES_WRITEREQUEST], callback,
                                    &UA_TYPES[UA_TYPES_WRITERESPONSE],
                                    userdata, requestId);
}

static UA_INLINE UA_StatusCode
UA_Client_AsyncService_call(UA_Client *client, const UA_CallRequest *request,
                            UA_ClientAsyncServiceCallback callback,
                            void *userdata, UA_UInt32 *requestId) {
    return __UA_Client_AsyncService(client, (const void*)request,
                                    &UA_TYPES[UA_TYPES_CALLREQUEST], callback,
                                    &UA_TYPES[UA_TYPES_CALLRESPONSE],
                                    userdata, requestId);
}

static UA_INLINE UA_StatusCode
UA_Client_AsyncService_browse(UA_Client *client, const UA_BrowseRequest *request,
                              UA_ClientAsyncServiceCallback callback,
                              void *userdata, UA_UInt32 *requestId) {
    return __UA_Client_AsyncService(client, (const void*)request,
                                    &UA_TYPES[UA_TYPES_BROWSEREQUEST], callback,
                                    &UA_TYPES[UA_TYPES_BROWSERESPONSE],
                                    userdata, requestId);
}
```

9.6.1 Highlevel Client Functionality

The following definitions are convenience functions making use of the standard OPC UA services in the background. This is a less flexible way of handling the stack, because at many places sensible defaults are presumed; at the same time using these functions is the easiest way of implementing an OPC UA application, as you will not have to consider all the details that go into the OPC UA services. If more flexibility is needed, you can always achieve the same functionality using the raw *OPC UA services*.

Read Attributes

The following functions can be used to retrieve a single node attribute. Use the regular service to read several attributes at once.

```c
/* Don't call this function, use the typed versions */
UA_StatusCode
__UA_Client_readAttribute(UA_Client *client, const UA_NodeId *nodeId,
                          UA_AttributeId attributeId, void *out,
                          const UA_DataType *outDataType);

static UA_INLINE UA_StatusCode
UA_Client_readNodeIdAttribute(UA_Client *client, const UA_NodeId nodeId,
                              UA_NodeId *outNodeId) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_NODEID,
                                     outNodeId, &UA_TYPES[UA_TYPES_NODEID]);
}

static UA_INLINE UA_StatusCode
UA_Client_readNodeClassAttribute(UA_Client *client, const UA_NodeId nodeId,
                                 UA_NodeClass *outNodeClass) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_NODECLASS,
                                     outNodeClass, &UA_TYPES[UA_TYPES_NODECLASS]);
}

static UA_INLINE UA_StatusCode
UA_Client_readBrowseNameAttribute(UA_Client *client, const UA_NodeId nodeId,
                                  UA_QualifiedName *outBrowseName) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_BROWSENAME,
                                     outBrowseName,
                                     &UA_TYPES[UA_TYPES_QUALIFIEDNAME]);
}

static UA_INLINE UA_StatusCode
UA_Client_readDisplayNameAttribute(UA_Client *client, const UA_NodeId nodeId,
                                   UA_LocalizedText *outDisplayName) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_DISPLAYNAME,
                                     outDisplayName,
                                     &UA_TYPES[UA_TYPES_LOCALIZEDTEXT]);
}

static UA_INLINE UA_StatusCode
UA_Client_readDescriptionAttribute(UA_Client *client, const UA_NodeId nodeId,
                                   UA_LocalizedText *outDescription) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_DESCRIPTION,
                                     outDescription,
                                     &UA_TYPES[UA_TYPES_LOCALIZEDTEXT]);
}

static UA_INLINE UA_StatusCode
UA_Client_readWriteMaskAttribute(UA_Client *client, const UA_NodeId nodeId,
                                 UA_UInt32 *outWriteMask) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_WRITEMASK,
```

```
                                      outWriteMask, &UA_TYPES[UA_TYPES_UINT32]);
}

static UA_INLINE UA_StatusCode
UA_Client_readUserWriteMaskAttribute(UA_Client *client, const UA_NodeId nodeId,
                                     UA_UInt32 *outUserWriteMask) {
    return __UA_Client_readAttribute(client, &nodeId,
                                     UA_ATTRIBUTEID_USERWRITEMASK,
                                     outUserWriteMask,
                                     &UA_TYPES[UA_TYPES_UINT32]);
}

static UA_INLINE UA_StatusCode
UA_Client_readIsAbstractAttribute(UA_Client *client, const UA_NodeId nodeId,
                                  UA_Boolean *outIsAbstract) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_ISABSTRACT,
                                     outIsAbstract, &UA_TYPES[UA_TYPES_BOOLEAN]);
}

static UA_INLINE UA_StatusCode
UA_Client_readSymmetricAttribute(UA_Client *client, const UA_NodeId nodeId,
                                 UA_Boolean *outSymmetric) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_SYMMETRIC,
                                     outSymmetric, &UA_TYPES[UA_TYPES_BOOLEAN]);
}

static UA_INLINE UA_StatusCode
UA_Client_readInverseNameAttribute(UA_Client *client, const UA_NodeId nodeId,
                                   UA_LocalizedText *outInverseName) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_INVERSENAME,
                                     outInverseName,
                                     &UA_TYPES[UA_TYPES_LOCALIZEDTEXT]);
}

static UA_INLINE UA_StatusCode
UA_Client_readContainsNoLoopsAttribute(UA_Client *client, const UA_NodeId nodeId,
                                       UA_Boolean *outContainsNoLoops) {
    return __UA_Client_readAttribute(client, &nodeId,
                                     UA_ATTRIBUTEID_CONTAINSNOLOOPS,
                                     outContainsNoLoops,
                                     &UA_TYPES[UA_TYPES_BOOLEAN]);
}

static UA_INLINE UA_StatusCode
UA_Client_readEventNotifierAttribute(UA_Client *client, const UA_NodeId nodeId,
                                     UA_Byte *outEventNotifier) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_EVENTNOTIFIER,
                                     outEventNotifier, &UA_TYPES[UA_TYPES_BYTE]);
}

static UA_INLINE UA_StatusCode
UA_Client_readValueAttribute(UA_Client *client, const UA_NodeId nodeId,
                             UA_Variant *outValue) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_VALUE,
                                     outValue, &UA_TYPES[UA_TYPES_VARIANT]);
}

static UA_INLINE UA_StatusCode
UA_Client_readDataTypeAttribute(UA_Client *client, const UA_NodeId nodeId,
                                UA_NodeId *outDataType) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_DATATYPE,
```

```
                                 outDataType, &UA_TYPES[UA_TYPES_NODEID]);
}

static UA_INLINE UA_StatusCode
UA_Client_readValueRankAttribute(UA_Client *client, const UA_NodeId nodeId,
                                 UA_Int32 *outValueRank) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_VALUERANK,
                                     outValueRank, &UA_TYPES[UA_TYPES_INT32]);
}

UA_StatusCode
UA_Client_readArrayDimensionsAttribute(UA_Client *client, const UA_NodeId nodeId,
                                       size_t *outArrayDimensionsSize,
                                       UA_UInt32 **outArrayDimensions);

static UA_INLINE UA_StatusCode
UA_Client_readAccessLevelAttribute(UA_Client *client, const UA_NodeId nodeId,
                                   UA_Byte *outAccessLevel) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_ACCESSLEVEL,
                                     outAccessLevel, &UA_TYPES[UA_TYPES_BYTE]);
}

static UA_INLINE UA_StatusCode
UA_Client_readUserAccessLevelAttribute(UA_Client *client, const UA_NodeId nodeId,
                                       UA_Byte *outUserAccessLevel) {
    return __UA_Client_readAttribute(client, &nodeId,
                                     UA_ATTRIBUTEID_USERACCESSLEVEL,
                                     outUserAccessLevel,
                                     &UA_TYPES[UA_TYPES_BYTE]);
}

static UA_INLINE UA_StatusCode
UA_Client_readMinimumSamplingIntervalAttribute(UA_Client *client,
                                               const UA_NodeId nodeId,
                                               UA_Double *outMinSamplingInterval) {
    return __UA_Client_readAttribute(client, &nodeId,
                                     UA_ATTRIBUTEID_MINIMUMSAMPLINGINTERVAL,
                                     outMinSamplingInterval,
                                     &UA_TYPES[UA_TYPES_DOUBLE]);
}

static UA_INLINE UA_StatusCode
UA_Client_readHistorizingAttribute(UA_Client *client, const UA_NodeId nodeId,
                                   UA_Boolean *outHistorizing) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_HISTORIZING,
                                     outHistorizing, &UA_TYPES[UA_TYPES_BOOLEAN]);
}

static UA_INLINE UA_StatusCode
UA_Client_readExecutableAttribute(UA_Client *client, const UA_NodeId nodeId,
                                  UA_Boolean *outExecutable) {
    return __UA_Client_readAttribute(client, &nodeId, UA_ATTRIBUTEID_EXECUTABLE,
                                     outExecutable, &UA_TYPES[UA_TYPES_BOOLEAN]);
}

static UA_INLINE UA_StatusCode
UA_Client_readUserExecutableAttribute(UA_Client *client, const UA_NodeId nodeId,
                                      UA_Boolean *outUserExecutable) {
    return __UA_Client_readAttribute(client, &nodeId,
                                     UA_ATTRIBUTEID_USEREXECUTABLE,
                                     outUserExecutable,
```

```
                            &UA_TYPES[UA_TYPES_BOOLEAN]);
}
```

Write Attributes

The following functions can be use to write a single node attribute at a time. Use the regular write service to write several attributes at once.

```c
/* Don't call this function, use the typed versions */
UA_StatusCode
__UA_Client_writeAttribute(UA_Client *client, const UA_NodeId *nodeId,
                           UA_AttributeId attributeId, const void *in,
                           const UA_DataType *inDataType);

static UA_INLINE UA_StatusCode
UA_Client_writeNodeIdAttribute(UA_Client *client, const UA_NodeId nodeId,
                               const UA_NodeId *newNodeId) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_NODEID,
                                      newNodeId, &UA_TYPES[UA_TYPES_NODEID]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeNodeClassAttribute(UA_Client *client, const UA_NodeId nodeId,
                                  const UA_NodeClass *newNodeClass) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_NODECLASS,
                                      newNodeClass, &UA_TYPES[UA_TYPES_NODECLASS]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeBrowseNameAttribute(UA_Client *client, const UA_NodeId nodeId,
                                   const UA_QualifiedName *newBrowseName) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_BROWSENAME,
                                      newBrowseName,
                                      &UA_TYPES[UA_TYPES_QUALIFIEDNAME]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeDisplayNameAttribute(UA_Client *client, const UA_NodeId nodeId,
                                    const UA_LocalizedText *newDisplayName) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_DISPLAYNAME,
                                      newDisplayName,
                                      &UA_TYPES[UA_TYPES_LOCALIZEDTEXT]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeDescriptionAttribute(UA_Client *client, const UA_NodeId nodeId,
                                    const UA_LocalizedText *newDescription) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_DESCRIPTION,
                                      newDescription,
                                      &UA_TYPES[UA_TYPES_LOCALIZEDTEXT]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeWriteMaskAttribute(UA_Client *client, const UA_NodeId nodeId,
                                  const UA_UInt32 *newWriteMask) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_WRITEMASK,
                                      newWriteMask, &UA_TYPES[UA_TYPES_UINT32]);
}

static UA_INLINE UA_StatusCode
```

```
UA_Client_writeUserWriteMaskAttribute(UA_Client *client, const UA_NodeId nodeId,
                                      const UA_UInt32 *newUserWriteMask) {
    return __UA_Client_writeAttribute(client, &nodeId,
                                      UA_ATTRIBUTEID_USERWRITEMASK,
                                      newUserWriteMask,
                                      &UA_TYPES[UA_TYPES_UINT32]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeIsAbstractAttribute(UA_Client *client, const UA_NodeId nodeId,
                                   const UA_Boolean *newIsAbstract) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_ISABSTRACT,
                                      newIsAbstract, &UA_TYPES[UA_TYPES_BOOLEAN]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeSymmetricAttribute(UA_Client *client, const UA_NodeId nodeId,
                                  const UA_Boolean *newSymmetric) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_SYMMETRIC,
                                      newSymmetric, &UA_TYPES[UA_TYPES_BOOLEAN]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeInverseNameAttribute(UA_Client *client, const UA_NodeId nodeId,
                                    const UA_LocalizedText *newInverseName) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_INVERSENAME,
                                      newInverseName,
                                      &UA_TYPES[UA_TYPES_LOCALIZEDTEXT]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeContainsNoLoopsAttribute(UA_Client *client, const UA_NodeId nodeId,
                                        const UA_Boolean *newContainsNoLoops) {
    return __UA_Client_writeAttribute(client, &nodeId,
                                      UA_ATTRIBUTEID_CONTAINSNOLOOPS,
                                      newContainsNoLoops,
                                      &UA_TYPES[UA_TYPES_BOOLEAN]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeEventNotifierAttribute(UA_Client *client, const UA_NodeId nodeId,
                                      const UA_Byte *newEventNotifier) {
    return __UA_Client_writeAttribute(client, &nodeId,
                                      UA_ATTRIBUTEID_EVENTNOTIFIER,
                                      newEventNotifier,
                                      &UA_TYPES[UA_TYPES_BYTE]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeValueAttribute(UA_Client *client, const UA_NodeId nodeId,
                              const UA_Variant *newValue) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_VALUE,
                                      newValue, &UA_TYPES[UA_TYPES_VARIANT]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeDataTypeAttribute(UA_Client *client, const UA_NodeId nodeId,
                                 const UA_NodeId *newDataType) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_DATATYPE,
                                      newDataType, &UA_TYPES[UA_TYPES_NODEID]);
}
```

```
static UA_INLINE UA_StatusCode
UA_Client_writeValueRankAttribute(UA_Client *client, const UA_NodeId nodeId,
                                  const UA_Int32 *newValueRank) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_VALUERANK,
                                      newValueRank, &UA_TYPES[UA_TYPES_INT32]);
}

UA_StatusCode
UA_Client_writeArrayDimensionsAttribute(UA_Client *client, const UA_NodeId nodeId,
                                        size_t newArrayDimensionsSize,
                                        const UA_UInt32 *newArrayDimensions);

static UA_INLINE UA_StatusCode
UA_Client_writeAccessLevelAttribute(UA_Client *client, const UA_NodeId nodeId,
                                    const UA_Byte *newAccessLevel) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_ACCESSLEVEL,
                                      newAccessLevel, &UA_TYPES[UA_TYPES_BYTE]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeUserAccessLevelAttribute(UA_Client *client, const UA_NodeId nodeId,
                                        const UA_Byte *newUserAccessLevel) {
    return __UA_Client_writeAttribute(client, &nodeId,
                                      UA_ATTRIBUTEID_USERACCESSLEVEL,
                                      newUserAccessLevel,
                                      &UA_TYPES[UA_TYPES_BYTE]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeMinimumSamplingIntervalAttribute(UA_Client *client,
                                                const UA_NodeId nodeId,
                                                const UA_Double *newMinInterval) {
    return __UA_Client_writeAttribute(client, &nodeId,
                                      UA_ATTRIBUTEID_MINIMUMSAMPLINGINTERVAL,
                                      newMinInterval, &UA_TYPES[UA_TYPES_DOUBLE]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeHistorizingAttribute(UA_Client *client, const UA_NodeId nodeId,
                                    const UA_Boolean *newHistorizing) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_HISTORIZING,
                                      newHistorizing, &UA_TYPES[UA_TYPES_BOOLEAN]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeExecutableAttribute(UA_Client *client, const UA_NodeId nodeId,
                                   const UA_Boolean *newExecutable) {
    return __UA_Client_writeAttribute(client, &nodeId, UA_ATTRIBUTEID_EXECUTABLE,
                                      newExecutable, &UA_TYPES[UA_TYPES_BOOLEAN]);
}

static UA_INLINE UA_StatusCode
UA_Client_writeUserExecutableAttribute(UA_Client *client, const UA_NodeId nodeId,
                                       const UA_Boolean *newUserExecutable) {
    return __UA_Client_writeAttribute(client, &nodeId,
                                      UA_ATTRIBUTEID_USEREXECUTABLE,
                                      newUserExecutable,
                                      &UA_TYPES[UA_TYPES_BOOLEAN]);
}
```

Method Calling

```
UA_StatusCode
UA_Client_call(UA_Client *client, const UA_NodeId objectId,
               const UA_NodeId methodId, size_t inputSize, const UA_Variant *input,
               size_t *outputSize, UA_Variant **output);
```

Node Management

See the section on *server-side node management*.

```
UA_StatusCode
UA_Client_addReference(UA_Client *client, const UA_NodeId sourceNodeId,
                       const UA_NodeId referenceTypeId, UA_Boolean isForward,
                       const UA_String targetServerUri,
                       const UA_ExpandedNodeId targetNodeId,
                       UA_NodeClass targetNodeClass);

UA_StatusCode
UA_Client_deleteReference(UA_Client *client, const UA_NodeId sourceNodeId,
                          const UA_NodeId referenceTypeId, UA_Boolean isForward,
                          const UA_ExpandedNodeId targetNodeId,
                          UA_Boolean deleteBidirectional);

UA_StatusCode
UA_Client_deleteNode(UA_Client *client, const UA_NodeId nodeId,
                     UA_Boolean deleteTargetReferences);

/* Protect against redundant definitions for server/client */
#ifndef UA_DEFAULT_ATTRIBUTES_DEFINED
#define UA_DEFAULT_ATTRIBUTES_DEFINED
/* The default for variables is "BaseDataType" for the datatype, -2 for the
 * valuerank and a read-accesslevel. */
extern const UA_VariableAttributes UA_VariableAttributes_default;
extern const UA_VariableTypeAttributes UA_VariableTypeAttributes_default;
/* Methods are executable by default */
extern const UA_MethodAttributes UA_MethodAttributes_default;
/* The remaining attribute definitions are currently all zeroed out */
extern const UA_ObjectAttributes UA_ObjectAttributes_default;
extern const UA_ObjectTypeAttributes UA_ObjectTypeAttributes_default;
extern const UA_ReferenceTypeAttributes UA_ReferenceTypeAttributes_default;
extern const UA_DataTypeAttributes UA_DataTypeAttributes_default;
extern const UA_ViewAttributes UA_ViewAttributes_default;
#endif

/* Don't call this function, use the typed versions */
UA_StatusCode
__UA_Client_addNode(UA_Client *client, const UA_NodeClass nodeClass,
                    const UA_NodeId requestedNewNodeId,
                    const UA_NodeId parentNodeId,
                    const UA_NodeId referenceTypeId,
                    const UA_QualifiedName browseName,
                    const UA_NodeId typeDefinition, const UA_NodeAttributes *attr,
                    const UA_DataType *attributeType, UA_NodeId *outNewNodeId);

static UA_INLINE UA_StatusCode
UA_Client_addVariableNode(UA_Client *client, const UA_NodeId requestedNewNodeId,
                          const UA_NodeId parentNodeId,
                          const UA_NodeId referenceTypeId,
                          const UA_QualifiedName browseName,
```

(continues on next page)

```
                             const UA_NodeId typeDefinition,
                             const UA_VariableAttributes attr,
                             UA_NodeId *outNewNodeId) {
    return __UA_Client_addNode(client, UA_NODECLASS_VARIABLE, requestedNewNodeId,
                               parentNodeId, referenceTypeId, browseName,
                               typeDefinition, (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_VARIABLEATTRIBUTES],
                               outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Client_addVariableTypeNode(UA_Client *client,
                              const UA_NodeId requestedNewNodeId,
                              const UA_NodeId parentNodeId,
                              const UA_NodeId referenceTypeId,
                              const UA_QualifiedName browseName,
                              const UA_VariableTypeAttributes attr,
                              UA_NodeId *outNewNodeId) {
    return __UA_Client_addNode(client, UA_NODECLASS_VARIABLETYPE,
                               requestedNewNodeId,
                               parentNodeId, referenceTypeId, browseName,
                               UA_NODEID_NULL, (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_VARIABLETYPEATTRIBUTES],
                               outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Client_addObjectNode(UA_Client *client, const UA_NodeId requestedNewNodeId,
                        const UA_NodeId parentNodeId,
                        const UA_NodeId referenceTypeId,
                        const UA_QualifiedName browseName,
                        const UA_NodeId typeDefinition,
                        const UA_ObjectAttributes attr, UA_NodeId *outNewNodeId) {
    return __UA_Client_addNode(client, UA_NODECLASS_OBJECT, requestedNewNodeId,
                               parentNodeId, referenceTypeId, browseName,
                               typeDefinition, (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_OBJECTATTRIBUTES], outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Client_addObjectTypeNode(UA_Client *client, const UA_NodeId requestedNewNodeId,
                            const UA_NodeId parentNodeId,
                            const UA_NodeId referenceTypeId,
                            const UA_QualifiedName browseName,
                            const UA_ObjectTypeAttributes attr,
                            UA_NodeId *outNewNodeId) {
    return __UA_Client_addNode(client, UA_NODECLASS_OBJECTTYPE, requestedNewNodeId,
                               parentNodeId, referenceTypeId, browseName,
                               UA_NODEID_NULL, (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_OBJECTTYPEATTRIBUTES],
                               outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Client_addViewNode(UA_Client *client, const UA_NodeId requestedNewNodeId,
                      const UA_NodeId parentNodeId,
                      const UA_NodeId referenceTypeId,
                      const UA_QualifiedName browseName,
                      const UA_ViewAttributes attr,
                      UA_NodeId *outNewNodeId) {
    return __UA_Client_addNode(client, UA_NODECLASS_VIEW, requestedNewNodeId,
```

```
                                parentNodeId, referenceTypeId, browseName,
                                UA_NODEID_NULL, (const UA_NodeAttributes*)&attr,
                                &UA_TYPES[UA_TYPES_VIEWATTRIBUTES], outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Client_addReferenceTypeNode(UA_Client *client,
                               const UA_NodeId requestedNewNodeId,
                               const UA_NodeId parentNodeId,
                               const UA_NodeId referenceTypeId,
                               const UA_QualifiedName browseName,
                               const UA_ReferenceTypeAttributes attr,
                               UA_NodeId *outNewNodeId) {
    return __UA_Client_addNode(client, UA_NODECLASS_REFERENCETYPE,
                               requestedNewNodeId,
                               parentNodeId, referenceTypeId, browseName,
                               UA_NODEID_NULL, (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_REFERENCETYPEATTRIBUTES],
                               outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Client_addDataTypeNode(UA_Client *client, const UA_NodeId requestedNewNodeId,
                          const UA_NodeId parentNodeId,
                          const UA_NodeId referenceTypeId,
                          const UA_QualifiedName browseName,
                          const UA_DataTypeAttributes attr,
                          UA_NodeId *outNewNodeId) {
    return __UA_Client_addNode(client, UA_NODECLASS_DATATYPE, requestedNewNodeId,
                               parentNodeId, referenceTypeId, browseName,
                               UA_NODEID_NULL, (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_DATATYPEATTRIBUTES],
                               outNewNodeId);
}

static UA_INLINE UA_StatusCode
UA_Client_addMethodNode(UA_Client *client, const UA_NodeId requestedNewNodeId,
                        const UA_NodeId parentNodeId,
                        const UA_NodeId referenceTypeId,
                        const UA_QualifiedName browseName,
                        const UA_MethodAttributes attr,
                        UA_NodeId *outNewNodeId) {
    return __UA_Client_addNode(client, UA_NODECLASS_METHOD, requestedNewNodeId,
                               parentNodeId, referenceTypeId, browseName,
                               UA_NODEID_NULL, (const UA_NodeAttributes*)&attr,
                               &UA_TYPES[UA_TYPES_METHODATTRIBUTES], outNewNodeId);
}
```

Misc Highlevel Functionality

```
/* Get the namespace-index of a namespace-URI
 *
 * @param client The UA_Client struct for this connection
 * @param namespaceUri The interested namespace URI
 * @param namespaceIndex The namespace index of the URI. The value is unchanged
 *        in case of an error
 * @return Indicates whether the operation succeeded or returns an error code */
UA_StatusCode
UA_Client_NamespaceGetIndex(UA_Client *client, UA_String *namespaceUri,
```

```
                            UA_UInt16 *namespaceIndex);

#ifndef HAVE_NODEITER_CALLBACK
#define HAVE_NODEITER_CALLBACK
/* Iterate over all nodes referenced by parentNodeId by calling the callback
   function for each child node */
typedef UA_StatusCode (*UA_NodeIteratorCallback)(UA_NodeId childId, UA_Boolean␣
→isInverse,
                                                 UA_NodeId referenceTypeId, void␣
→*handle);
#endif

UA_StatusCode
UA_Client_forEachChildNodeCall(UA_Client *client, UA_NodeId parentNodeId,
                               UA_NodeIteratorCallback callback, void *handle) ;
```

9.6.2 Subscriptions

Subscriptions in OPC UA are asynchronous. That is, the client sends several PublishRequests to the server. The server returns PublishResponses with notifications. But only when a notification has been generated. The client does not wait for the responses and continues normal operations.

Note the difference between Subscriptions and MonitoredItems. Subscriptions are used to report back notifications. MonitoredItems are used to generate notifications. Every MonitoredItem is attached to exactly one Subscription. And a Subscription can contain many MonitoredItems.

The client automatically processes PublishResponses (with a callback) in the background and keeps enough PublishRequests in transit. The PublishResponses may be recieved during a synchronous service call or in UA_Client_runAsync.

```
/* Callbacks defined for Subscriptions */
typedef void (*UA_Client_DeleteSubscriptionCallback)
    (UA_Client *client, UA_UInt32 subId, void *subContext);

typedef void (*UA_Client_StatusChangeNotificationCallback)
    (UA_Client *client, UA_UInt32 subId, void *subContext,
     UA_StatusChangeNotification *notification);

/* Provides default values for a new subscription.
 *
 * RequestedPublishingInterval:  500.0 [ms]
 * RequestedLifetimeCount: 10000
 * RequestedMaxKeepAliveCount: 10
 * MaxNotificationsPerPublish: 0 (unlimited)
 * PublishingEnabled: true
 * Priority: 0 */
static UA_INLINE UA_CreateSubscriptionRequest
UA_CreateSubscriptionRequest_default(void) {
    UA_CreateSubscriptionRequest request;
    UA_CreateSubscriptionRequest_init(&request);

    request.requestedPublishingInterval = 500.0;
    request.requestedLifetimeCount = 10000;
    request.requestedMaxKeepAliveCount = 10;
    request.maxNotificationsPerPublish = 0;
    request.publishingEnabled = true;
    request.priority = 0;
    return request;
}
```

```
UA_CreateSubscriptionResponse
UA_Client_Subscriptions_create(UA_Client *client,
                               const UA_CreateSubscriptionRequest request,
                               void *subscriptionContext,
                               UA_Client_StatusChangeNotificationCallback␣
→statusChangeCallback,
                               UA_Client_DeleteSubscriptionCallback␣
→deleteCallback);

UA_ModifySubscriptionResponse
UA_Client_Subscriptions_modify(UA_Client *client, const UA_
→ModifySubscriptionRequest request);

UA_DeleteSubscriptionsResponse
UA_Client_Subscriptions_delete(UA_Client *client,
                               const UA_DeleteSubscriptionsRequest request);

/* Delete a single subscription */
UA_StatusCode
UA_Client_Subscriptions_deleteSingle(UA_Client *client, UA_UInt32 subscriptionId);

static UA_INLINE UA_SetPublishingModeResponse
UA_Client_Subscriptions_setPublishingMode(UA_Client *client,
                                          const UA_SetPublishingModeRequest␣
→request) {
    UA_SetPublishingModeResponse response;
    __UA_Client_Service(client, &request,
                        &UA_TYPES[UA_TYPES_SETPUBLISHINGMODEREQUEST], &response,
                        &UA_TYPES[UA_TYPES_SETPUBLISHINGMODERESPONSE]);
    return response;
}
```

9.6.3 MonitoredItems

MonitoredItems for Events indicate the `EventNotifier` attribute. This indicates to the server not to monitor changes of the attribute, but to forward Event notifications from that node.

During the creation of a MonitoredItem, the server may return changed adjusted parameters. Use `UA_Client_MonitoredItem_getParameters` to get the current parameters.

```
/* Provides default values for a new monitored item. */
static UA_INLINE UA_MonitoredItemCreateRequest
UA_MonitoredItemCreateRequest_default(UA_NodeId nodeId) {
    UA_MonitoredItemCreateRequest request;
    UA_MonitoredItemCreateRequest_init(&request);
    request.itemToMonitor.nodeId = nodeId;
    request.itemToMonitor.attributeId = UA_ATTRIBUTEID_VALUE;
    request.monitoringMode = UA_MONITORINGMODE_REPORTING;
    request.requestedParameters.samplingInterval = 250;
    request.requestedParameters.discardOldest = true;
    request.requestedParameters.queueSize = 1;
    return request;
}

/* Callback for the deletion of a MonitoredItem */
typedef void (*UA_Client_DeleteMonitoredItemCallback)
    (UA_Client *client, UA_UInt32 subId, void *subContext,
     UA_UInt32 monId, void *monContext);

/* Callback for DataChange notifications */
```

```c
typedef void (*UA_Client_DataChangeNotificationCallback)
    (UA_Client *client, UA_UInt32 subId, void *subContext,
     UA_UInt32 monId, void *monContext,
     UA_DataValue *value);

/* Callback for Event notifications */
typedef void (*UA_Client_EventNotificationCallback)
    (UA_Client *client, UA_UInt32 subId, void *subContext,
     UA_UInt32 monId, void *monContext,
     size_t nEventFields, UA_Variant *eventFields);

/* Don't use to monitor the EventNotifier attribute */
UA_CreateMonitoredItemsResponse
UA_Client_MonitoredItems_createDataChanges(UA_Client *client,
            const UA_CreateMonitoredItemsRequest request, void **contexts,
            UA_Client_DataChangeNotificationCallback *callbacks,
            UA_Client_DeleteMonitoredItemCallback *deleteCallbacks);

UA_MonitoredItemCreateResult
UA_Client_MonitoredItems_createDataChange(UA_Client *client, UA_UInt32␣
→subscriptionId,
         UA_TimestampsToReturn timestampsToReturn, const UA_
→MonitoredItemCreateRequest item,
         void *context, UA_Client_DataChangeNotificationCallback callback,
         UA_Client_DeleteMonitoredItemCallback deleteCallback);

/* Monitor the EventNotifier attribute only */
UA_CreateMonitoredItemsResponse
UA_Client_MonitoredItems_createEvents(UA_Client *client,
            const UA_CreateMonitoredItemsRequest request, void **contexts,
            UA_Client_EventNotificationCallback *callbacks,
            UA_Client_DeleteMonitoredItemCallback *deleteCallback);

UA_MonitoredItemCreateResult
UA_Client_MonitoredItems_createEvent(UA_Client *client, UA_UInt32 subscriptionId,
         UA_TimestampsToReturn timestampsToReturn, const UA_
→MonitoredItemCreateRequest item,
         void *context, UA_Client_EventNotificationCallback callback,
         UA_Client_DeleteMonitoredItemCallback deleteCallback);

UA_DeleteMonitoredItemsResponse
UA_Client_MonitoredItems_delete(UA_Client *client, const UA_
→DeleteMonitoredItemsRequest);

UA_StatusCode
UA_Client_MonitoredItems_deleteSingle(UA_Client *client, UA_UInt32 subscriptionId,␣
→UA_UInt32 monitoredItemId);
```

The following service calls go directly to the server. The MonitoredItem settings are not stored in the client.

```c
static UA_INLINE UA_ModifyMonitoredItemsResponse
UA_Client_MonitoredItems_modify(UA_Client *client,
                                const UA_ModifyMonitoredItemsRequest request) {
    UA_ModifyMonitoredItemsResponse response;
    __UA_Client_Service(client,
                        &request, &UA_TYPES[UA_TYPES_MODIFYMONITOREDITEMSREQUEST],
                        &response, &UA_TYPES[UA_TYPES_
→MODIFYMONITOREDITEMSRESPONSE]);
    return response;
}
```

```
static UA_INLINE UA_SetMonitoringModeResponse
UA_Client_MonitoredItems_setMonitoringMode(UA_Client *client,
                                           const UA_SetMonitoringModeRequest
→request) {
    UA_SetMonitoringModeResponse response;
    __UA_Client_Service(client,
                        &request, &UA_TYPES[UA_TYPES_SETMONITORINGMODEREQUEST],
                        &response, &UA_TYPES[UA_TYPES_SETMONITORINGMODERESPONSE]);
    return response;
}


static UA_INLINE UA_SetTriggeringResponse
UA_Client_MonitoredItems_setTriggering(UA_Client *client,
                                       const UA_SetTriggeringRequest request) {
    UA_SetTriggeringResponse response;
    __UA_Client_Service(client,
                        &request, &UA_TYPES[UA_TYPES_SETTRIGGERINGREQUEST],
                        &response, &UA_TYPES[UA_TYPES_SETTRIGGERINGRESPONSE]);
    return response;
}
```

9.6.4 Deprecated API

The following API is kept for backwards compatibility. It will be removed in future releases.

```
typedef struct {
    UA_Double requestedPublishingInterval;
    UA_UInt32 requestedLifetimeCount;
    UA_UInt32 requestedMaxKeepAliveCount;
    UA_UInt32 maxNotificationsPerPublish;
    UA_Boolean publishingEnabled;
    UA_Byte priority;
} UA_SubscriptionSettings;

extern const UA_SubscriptionSettings UA_SubscriptionSettings_default;

UA_DEPRECATED UA_StatusCode
UA_Client_Subscriptions_new(UA_Client *client, UA_SubscriptionSettings settings,
                            UA_UInt32 *newSubscriptionId);

UA_DEPRECATED UA_StatusCode
UA_Client_Subscriptions_remove(UA_Client *client, UA_UInt32 subscriptionId);

/* Send a publish request and wait until a response to the request is processed.
 * Note that other publish responses may be processed in the background until
 * then. */
UA_DEPRECATED UA_StatusCode
UA_Client_Subscriptions_manuallySendPublishRequest(UA_Client *client);


/* For monitoring DataChanges */
typedef void (*UA_MonitoredItemHandlingFunction)(UA_Client *client, UA_UInt32
→monId,
                                                 UA_DataValue *value, void
→*context);

UA_DEPRECATED UA_StatusCode
UA_Client_Subscriptions_addMonitoredItems(UA_Client *client, const UA_UInt32
→subscriptionId,
                                          UA_MonitoredItemCreateRequest *items,
→size_t itemsSize,
```

```
                                             UA_MonitoredItemHandlingFunction *hfs,
                                             void **hfContexts, UA_StatusCode
↪*itemResults,
                                             UA_UInt32 *newMonitoredItemIds);

UA_DEPRECATED UA_StatusCode
UA_Client_Subscriptions_addMonitoredItem(UA_Client *client, UA_UInt32
↪subscriptionId,
                                             UA_NodeId nodeId, UA_UInt32 attributeID,
                                             UA_MonitoredItemHandlingFunction hf,
                                             void *hfContext,
                                             UA_UInt32 *newMonitoredItemId,
                                             UA_Double samplingInterval);

/* Monitored Events have different payloads from DataChanges. So they use a
 * different callback method signature. */
typedef void (*UA_MonitoredEventHandlingFunction)(UA_Client *client,
                                             const UA_UInt32 monId,
                                             const size_t nEventFields,
                                             const UA_Variant *eventFields,
                                             void *context);

UA_DEPRECATED UA_StatusCode
UA_Client_Subscriptions_addMonitoredEvents(UA_Client *client, const UA_UInt32
↪subscriptionId,
                                             UA_MonitoredItemCreateRequest *items,
↪size_t itemsSize,
                                             UA_MonitoredEventHandlingFunction *hfs,
                                             void **hfContexts, UA_StatusCode
↪*itemResults,
                                             UA_UInt32 *newMonitoredItemIds);

/* TODO for 0.4: attribute is fix for events. */
UA_DEPRECATED UA_StatusCode
UA_Client_Subscriptions_addMonitoredEvent(UA_Client *client, UA_UInt32
↪subscriptionId,
                                             const UA_NodeId nodeId, UA_UInt32
↪attributeID,
                                             const UA_SimpleAttributeOperand
↪*selectClauses,
                                             size_t selectClausesSize,
                                             const UA_ContentFilterElement
↪*whereClauses,
                                             size_t whereClausesSize,
                                             const UA_MonitoredEventHandlingFunction
↪hf,
                                             void *hfContext, UA_UInt32
↪*newMonitoredItemId);

UA_DEPRECATED UA_StatusCode
UA_Client_Subscriptions_removeMonitoredItem(UA_Client *client, UA_UInt32
↪subscriptionId,
                                             UA_UInt32 monitoredItemId);

UA_DEPRECATED UA_StatusCode
UA_Client_Subscriptions_removeMonitoredItems(UA_Client *client, UA_UInt32
↪subscriptionId,
                                             UA_UInt32 *monitoredItemIds, size_t
↪itemsSize,
                                             UA_StatusCode *itemResults);
```

```
#endif
```

Standard-Defined Constants

This section contains numerical and string constants that are defined in the OPC UA standard.

10.1 Attribute Id

Every node in an OPC UA information model contains attributes depending on the node type. Possible attributes are as follows:

```
typedef enum {
    UA_ATTRIBUTEID_NODEID                 = 1,
    UA_ATTRIBUTEID_NODECLASS              = 2,
    UA_ATTRIBUTEID_BROWSENAME             = 3,
    UA_ATTRIBUTEID_DISPLAYNAME            = 4,
    UA_ATTRIBUTEID_DESCRIPTION            = 5,
    UA_ATTRIBUTEID_WRITEMASK              = 6,
    UA_ATTRIBUTEID_USERWRITEMASK          = 7,
    UA_ATTRIBUTEID_ISABSTRACT             = 8,
    UA_ATTRIBUTEID_SYMMETRIC              = 9,
    UA_ATTRIBUTEID_INVERSENAME            = 10,
    UA_ATTRIBUTEID_CONTAINSNOLOOPS        = 11,
    UA_ATTRIBUTEID_EVENTNOTIFIER          = 12,
    UA_ATTRIBUTEID_VALUE                  = 13,
    UA_ATTRIBUTEID_DATATYPE               = 14,
    UA_ATTRIBUTEID_VALUERANK              = 15,
    UA_ATTRIBUTEID_ARRAYDIMENSIONS        = 16,
    UA_ATTRIBUTEID_ACCESSLEVEL            = 17,
    UA_ATTRIBUTEID_USERACCESSLEVEL        = 18,
    UA_ATTRIBUTEID_MINIMUMSAMPLINGINTERVAL = 19,
    UA_ATTRIBUTEID_HISTORIZING            = 20,
    UA_ATTRIBUTEID_EXECUTABLE             = 21,
    UA_ATTRIBUTEID_USEREXECUTABLE         = 22
} UA_AttributeId;
```

10.2 Access Level Masks

The access level to a node is given by the following constants that are ANDed with the overall access level.

```
#define UA_ACCESSLEVELMASK_READ          (0x01<<0)
#define UA_ACCESSLEVELMASK_WRITE         (0x01<<1)
#define UA_ACCESSLEVELMASK_HISTORYREAD   (0x01<<2)
#define UA_ACCESSLEVELMASK_HISTORYWRITE  (0x01<<3)
#define UA_ACCESSLEVELMASK_SEMANTICCHANGE (0x01<<4)
#define UA_ACCESSLEVELMASK_STATUSWRITE   (0x01<<5)
#define UA_ACCESSLEVELMASK_TIMESTAMPWRITE (0x01<<6)
```

10.3 Write Masks

The write mask and user write mask is given by the following constants that are ANDed for the overall write mask.
Part 3: 5.2.7 Table 2

```
#define UA_WRITEMASK_ACCESSLEVEL           (0x01<<0)
#define UA_WRITEMASK_ARRRAYDIMENSIONS      (0x01<<1)
#define UA_WRITEMASK_BROWSENAME            (0x01<<2)
#define UA_WRITEMASK_CONTAINSNOLOOPS       (0x01<<3)
#define UA_WRITEMASK_DATATYPE              (0x01<<4)
#define UA_WRITEMASK_DESCRIPTION           (0x01<<5)
#define UA_WRITEMASK_DISPLAYNAME           (0x01<<6)
#define UA_WRITEMASK_EVENTNOTIFIER         (0x01<<7)
#define UA_WRITEMASK_EXECUTABLE            (0x01<<8)
#define UA_WRITEMASK_HISTORIZING           (0x01<<9)
#define UA_WRITEMASK_INVERSENAME           (0x01<<10)
#define UA_WRITEMASK_ISABSTRACT            (0x01<<11)
#define UA_WRITEMASK_MINIMUMSAMPLINGINTERVAL (0x01<<12)
#define UA_WRITEMASK_NODECLASS             (0x01<<13)
#define UA_WRITEMASK_NODEID                (0x01<<14)
#define UA_WRITEMASK_SYMMETRIC             (0x01<<15)
#define UA_WRITEMASK_USERACCESSLEVEL       (0x01<<16)
#define UA_WRITEMASK_USEREXECUTABLE        (0x01<<17)
#define UA_WRITEMASK_USERWRITEMASK         (0x01<<18)
#define UA_WRITEMASK_VALUERANK             (0x01<<19)
#define UA_WRITEMASK_WRITEMASK             (0x01<<20)
#define UA_WRITEMASK_VALUEFORVARIABLETYPE  (0x01<<21)
```

10.4 StatusCodes

StatusCodes are extensively used in the OPC UA protocol and in the open62541 API. They are represented by the
StatusCode data type. The following definitions are autogenerated from the `Opc.Ua.StatusCodes.csv` file
provided with the OPC UA standard.

```
#define UA_STATUSCODE_GOOD 0x00
#define UA_STATUSCODE_BADUNEXPECTEDERROR 0x80010000 // An unexpected error
→occurred.
#define UA_STATUSCODE_BADINTERNALERROR 0x80020000 // An internal error occurred as
→a result of a programming or configuration error.
#define UA_STATUSCODE_BADOUTOFMEMORY 0x80030000 // Not enough memory to complete
→the operation.
#define UA_STATUSCODE_BADRESOURCEUNAVAILABLE 0x80040000 // An operating system
→resource is not available.
#define UA_STATUSCODE_BADCOMMUNICATIONERROR 0x80050000 // A low level
→communication error occurred.
#define UA_STATUSCODE_BADENCODINGERROR 0x80060000 // Encoding halted because of
→invalid data in the objects being serialized.
#define UA_STATUSCODE_BADDECODINGERROR 0x80070000 // Decoding halted because of
→invalid data in the stream.
```

```
#define UA_STATUSCODE_BADENCODINGLIMITSEXCEEDED 0x80080000 // The message encoding/
→decoding limits imposed by the stack have been exceeded.
#define UA_STATUSCODE_BADREQUESTTOOLARGE 0x80b80000 // The request message size␣
→exceeds limits set by the server.
#define UA_STATUSCODE_BADRESPONSETOOLARGE 0x80b90000 // The response message size␣
→exceeds limits set by the client.
#define UA_STATUSCODE_BADUNKNOWNRESPONSE 0x80090000 // An unrecognized response␣
→was received from the server.
#define UA_STATUSCODE_BADTIMEOUT 0x800a0000 // The operation timed out.
#define UA_STATUSCODE_BADSERVICEUNSUPPORTED 0x800b0000 // The server does not␣
→support the requested service.
#define UA_STATUSCODE_BADSHUTDOWN 0x800c0000 // The operation was cancelled␣
→because the application is shutting down.
#define UA_STATUSCODE_BADSERVERNOTCONNECTED 0x800d0000 // The operation could not␣
→complete because the client is not connected to the server.
#define UA_STATUSCODE_BADSERVERHALTED 0x800e0000 // The server has stopped and␣
→cannot process any requests.
#define UA_STATUSCODE_BADNOTHINGTODO 0x800f0000 // There was nothing to do because␣
→the client passed a list of operations with no elements.
#define UA_STATUSCODE_BADTOOMANYOPERATIONS 0x80100000 // The request could not be␣
→processed because it specified too many operations.
#define UA_STATUSCODE_BADTOOMANYMONITOREDITEMS 0x80db0000 // The request could not␣
→be processed because there are too many monitored items in the subscription.
#define UA_STATUSCODE_BADDATATYPEIDUNKNOWN 0x80110000 // The extension object␣
→cannot be (de)serialized because the data type id is not recognized.
#define UA_STATUSCODE_BADCERTIFICATEINVALID 0x80120000 // The certificate provided␣
→as a parameter is not valid.
#define UA_STATUSCODE_BADSECURITYCHECKSFAILED 0x80130000 // An error occurred␣
→verifying security.
#define UA_STATUSCODE_BADCERTIFICATETIMEINVALID 0x80140000 // The Certificate has␣
→expired or is not yet valid.
#define UA_STATUSCODE_BADCERTIFICATEISSUERTIMEINVALID 0x80150000 // An Issuer␣
→Certificate has expired or is not yet valid.
#define UA_STATUSCODE_BADCERTIFICATEHOSTNAMEINVALID 0x80160000 // The HostName␣
→used to connect to a Server does not match a HostName in the Certificate.
#define UA_STATUSCODE_BADCERTIFICATEURIINVALID 0x80170000 // The URI specified in␣
→the ApplicationDescription does not match the URI in the Certificate.
#define UA_STATUSCODE_BADCERTIFICATEUSENOTALLOWED 0x80180000 // The Certificate␣
→may not be used for the requested operation.
#define UA_STATUSCODE_BADCERTIFICATEISSUERUSENOTALLOWED 0x80190000 // The Issuer␣
→Certificate may not be used for the requested operation.
#define UA_STATUSCODE_BADCERTIFICATEUNTRUSTED 0x801a0000 // The Certificate is not␣
→trusted.
#define UA_STATUSCODE_BADCERTIFICATEREVOCATIONUNKNOWN 0x801b0000 // It was not␣
→possible to determine if the Certificate has been revoked.
#define UA_STATUSCODE_BADCERTIFICATEISSUERREVOCATIONUNKNOWN 0x801c0000 // It was␣
→not possible to determine if the Issuer Certificate has been revoked.
#define UA_STATUSCODE_BADCERTIFICATEREVOKED 0x801d0000 // The certificate has been␣
→revoked.
#define UA_STATUSCODE_BADCERTIFICATEISSUERREVOKED 0x801e0000 // The issuer␣
→certificate has been revoked.
#define UA_STATUSCODE_BADCERTIFICATECHAININCOMPLETE 0x810d0000 // The certificate␣
→chain is incomplete.
#define UA_STATUSCODE_BADUSERACCESSDENIED 0x801f0000 // User does not have␣
→permission to perform the requested operation.
#define UA_STATUSCODE_BADIDENTITYTOKENINVALID 0x80200000 // The user identity␣
→token is not valid.
#define UA_STATUSCODE_BADIDENTITYTOKENREJECTED 0x80210000 // The user identity␣
→token is valid but the server has rejected it.
#define UA_STATUSCODE_BADSECURECHANNELIDINVALID 0x80220000 // The specified secure␣
→channel is no longer valid.
```

```
#define UA_STATUSCODE_BADINVALIDTIMESTAMP 0x80230000 // The timestamp is outside␣
↪the range allowed by the server.
#define UA_STATUSCODE_BADNONCEINVALID 0x80240000 // The nonce does appear to be␣
↪not a random value or it is not the correct length.
#define UA_STATUSCODE_BADSESSIONIDINVALID 0x80250000 // The session id is not␣
↪valid.
#define UA_STATUSCODE_BADSESSIONCLOSED 0x80260000 // The session was closed by the␣
↪client.
#define UA_STATUSCODE_BADSESSIONNOTACTIVATED 0x80270000 // The session cannot be␣
↪used because ActivateSession has not been called.
#define UA_STATUSCODE_BADSUBSCRIPTIONIDINVALID 0x80280000 // The subscription id␣
↪is not valid.
#define UA_STATUSCODE_BADREQUESTHEADERINVALID 0x802a0000 // The header for the␣
↪request is missing or invalid.
#define UA_STATUSCODE_BADTIMESTAMPSTORETURNINVALID 0x802b0000 // The timestamps to␣
↪return parameter is invalid.
#define UA_STATUSCODE_BADREQUESTCANCELLEDBYCLIENT 0x802c0000 // The request was␣
↪cancelled by the client.
#define UA_STATUSCODE_BADTOOMANYARGUMENTS 0x80e50000 // Too many arguments were␣
↪provided.
#define UA_STATUSCODE_BADLICENSEEXPIRED 0x810E0000 // The server requires a␣
↪license to operate in general or to perform a service or operation, but existing␣
↪license is expired.
#define UA_STATUSCODE_BADLICENSELIMITSEXCEEDED 0x810F0000 // The server has limits␣
↪on number of allowed operations / objects, based on installed licenses, and␣
↪these limits where exceeded.
#define UA_STATUSCODE_BADLICENSENOTAVAILABLE 0x81100000 // The server does not␣
↪have a license which is required to operate in general or to perform a service␣
↪or operation.
#define UA_STATUSCODE_GOODSUBSCRIPTIONTRANSFERRED 0x002d0000 // The subscription␣
↪was transferred to another session.
#define UA_STATUSCODE_GOODCOMPLETESASYNCHRONOUSLY 0x002e0000 // The processing␣
↪will complete asynchronously.
#define UA_STATUSCODE_GOODOVERLOAD 0x002f0000 // Sampling has slowed down due to␣
↪resource limitations.
#define UA_STATUSCODE_GOODCLAMPED 0x00300000 // The value written was accepted but␣
↪was clamped.
#define UA_STATUSCODE_BADNOCOMMUNICATION 0x80310000 // Communication with the data␣
↪source is defined, but not established, and there is no last known value␣
↪available.
#define UA_STATUSCODE_BADWAITINGFORINITIALDATA 0x80320000 // Waiting for the␣
↪server to obtain values from the underlying data source.
#define UA_STATUSCODE_BADNODEIDINVALID 0x80330000 // The syntax of the node id is␣
↪not valid.
#define UA_STATUSCODE_BADNODEIDUNKNOWN 0x80340000 // The node id refers to a node␣
↪that does not exist in the server address space.
#define UA_STATUSCODE_BADATTRIBUTEIDINVALID 0x80350000 // The attribute is not␣
↪supported for the specified Node.
#define UA_STATUSCODE_BADINDEXRANGEINVALID 0x80360000 // The syntax of the index␣
↪range parameter is invalid.
#define UA_STATUSCODE_BADINDEXRANGENODATA 0x80370000 // No data exists within the␣
↪range of indexes specified.
#define UA_STATUSCODE_BADDATAENCODINGINVALID 0x80380000 // The data encoding is␣
↪invalid.
#define UA_STATUSCODE_BADDATAENCODINGUNSUPPORTED 0x80390000 // The server does not␣
↪support the requested data encoding for the node.
#define UA_STATUSCODE_BADNOTREADABLE 0x803a0000 // The access level does not allow␣
↪reading or subscribing to the Node.
#define UA_STATUSCODE_BADNOTWRITABLE 0x803b0000 // The access level does not allow␣
↪writing to the Node.
#define UA_STATUSCODE_BADOUTOFRANGE 0x803c0000 // The value was out of range.
```

```
#define UA_STATUSCODE_BADNOTSUPPORTED 0x803d0000 // The requested operation is not␣
↪supported.
#define UA_STATUSCODE_BADNOTFOUND 0x803e0000 // A requested item was not found or␣
↪a search operation ended without success.
#define UA_STATUSCODE_BADOBJECTDELETED 0x803f0000 // The object cannot be used␣
↪because it has been deleted.
#define UA_STATUSCODE_BADNOTIMPLEMENTED 0x80400000 // Requested operation is not␣
↪implemented.
#define UA_STATUSCODE_BADMONITORINGMODEINVALID 0x80410000 // The monitoring mode␣
↪is invalid.
#define UA_STATUSCODE_BADMONITOREDITEMIDINVALID 0x80420000 // The monitoring item␣
↪id does not refer to a valid monitored item.
#define UA_STATUSCODE_BADMONITOREDITEMFILTERINVALID 0x80430000 // The monitored␣
↪item filter parameter is not valid.
#define UA_STATUSCODE_BADMONITOREDITEMFILTERUNSUPPORTED 0x80440000 // The server␣
↪does not support the requested monitored item filter.
#define UA_STATUSCODE_BADFILTERNOTALLOWED 0x80450000 // A monitoring filter cannot␣
↪be used in combination with the attribute specified.
#define UA_STATUSCODE_BADSTRUCTUREMISSING 0x80460000 // A mandatory structured␣
↪parameter was missing or null.
#define UA_STATUSCODE_BADEVENTFILTERINVALID 0x80470000 // The event filter is not␣
↪valid.
#define UA_STATUSCODE_BADCONTENTFILTERINVALID 0x80480000 // The content filter is␣
↪not valid.
#define UA_STATUSCODE_BADFILTEROPERATORINVALID 0x80c10000 // An unregognized␣
↪operator was provided in a filter.
#define UA_STATUSCODE_BADFILTEROPERATORUNSUPPORTED 0x80c20000 // A valid operator␣
↪was provided, but the server does not provide support for this filter operator.
#define UA_STATUSCODE_BADFILTEROPERANDCOUNTMISMATCH 0x80c30000 // The number of␣
↪operands provided for the filter operator was less then expected for the operand␣
↪provided.
#define UA_STATUSCODE_BADFILTEROPERANDINVALID 0x80490000 // The operand used in a␣
↪content filter is not valid.
#define UA_STATUSCODE_BADFILTERELEMENTINVALID 0x80c40000 // The referenced element␣
↪is not a valid element in the content filter.
#define UA_STATUSCODE_BADFILTERLITERALINVALID 0x80c50000 // The referenced literal␣
↪is not a valid value.
#define UA_STATUSCODE_BADCONTINUATIONPOINTINVALID 0x804a0000 // The continuation␣
↪point provide is longer valid.
#define UA_STATUSCODE_BADNOCONTINUATIONPOINTS 0x804b0000 // The operation could␣
↪not be processed because all continuation points have been allocated.
#define UA_STATUSCODE_BADREFERENCETYPEIDINVALID 0x804c0000 // The operation could␣
↪not be processed because all continuation points have been allocated.
#define UA_STATUSCODE_BADBROWSEDIRECTIONINVALID 0x804d0000 // The browse direction␣
↪is not valid.
#define UA_STATUSCODE_BADNODENOTINVIEW 0x804e0000 // The node is not part of the␣
↪view.
#define UA_STATUSCODE_BADSERVERURIINVALID 0x804f0000 // The ServerUri is not a␣
↪valid URI.
#define UA_STATUSCODE_BADSERVERNAMEMISSING 0x80500000 // No ServerName was␣
↪specified.
#define UA_STATUSCODE_BADDISCOVERYURLMISSING 0x80510000 // No DiscoveryUrl was␣
↪specified.
#define UA_STATUSCODE_BADSEMPAHOREFILEMISSING 0x80520000 // The semaphore file␣
↪specified by the client is not valid.
#define UA_STATUSCODE_BADREQUESTTYPEINVALID 0x80530000 // The security token␣
↪request type is not valid.
#define UA_STATUSCODE_BADSECURITYMODEREJECTED 0x80540000 // The security mode does␣
↪not meet the requirements set by the Server.
#define UA_STATUSCODE_BADSECURITYPOLICYREJECTED 0x80550000 // The security policy␣
↪does not meet the requirements set by the Server.
```

```
#define UA_STATUSCODE_BADTOOMANYSESSIONS 0x80560000 // The server has reached its␣
↪maximum number of sessions.
#define UA_STATUSCODE_BADUSERSIGNATUREINVALID 0x80570000 // The user token␣
↪signature is missing or invalid.
#define UA_STATUSCODE_BADAPPLICATIONSIGNATUREINVALID 0x80580000 // The signature␣
↪generated with the client certificate is missing or invalid.
#define UA_STATUSCODE_BADNOVALIDCERTIFICATES 0x80590000 // The client did not␣
↪provide at least one software certificate that is valid and meets the profile␣
↪requirements for the server.
#define UA_STATUSCODE_BADIDENTITYCHANGENOTSUPPORTED 0x80c60000 // The Server does␣
↪not support changing the user identity assigned to the session.
#define UA_STATUSCODE_BADREQUESTCANCELLEDBYREQUEST 0x805a0000 // The request was␣
↪cancelled by the client with the Cancel service.
#define UA_STATUSCODE_BADPARENTNODEIDINVALID 0x805b0000 // The parent node id does␣
↪not to refer to a valid node.
#define UA_STATUSCODE_BADREFERENCENOTALLOWED 0x805c0000 // The reference could not␣
↪be created because it violates constraints imposed by the data model.
#define UA_STATUSCODE_BADNODEIDREJECTED 0x805d0000 // The requested node id was␣
↪reject because it was either invalid or server does not allow node ids to be␣
↪specified by the client.
#define UA_STATUSCODE_BADNODEIDEXISTS 0x805e0000 // The requested node id is␣
↪already used by another node.
#define UA_STATUSCODE_BADNODECLASSINVALID 0x805f0000 // The node class is not␣
↪valid.
#define UA_STATUSCODE_BADBROWSENAMEINVALID 0x80600000 // The browse name is␣
↪invalid.
#define UA_STATUSCODE_BADBROWSENAMEDUPLICATED 0x80610000 // The browse name is not␣
↪unique among nodes that share the same relationship with the parent.
#define UA_STATUSCODE_BADNODEATTRIBUTESINVALID 0x80620000 // The node attributes␣
↪are not valid for the node class.
#define UA_STATUSCODE_BADTYPEDEFINITIONINVALID 0x80630000 // The type definition␣
↪node id does not reference an appropriate type node.
#define UA_STATUSCODE_BADSOURCENODEIDINVALID 0x80640000 // The source node id does␣
↪not reference a valid node.
#define UA_STATUSCODE_BADTARGETNODEIDINVALID 0x80650000 // The target node id does␣
↪not reference a valid node.
#define UA_STATUSCODE_BADDUPLICATEREFERENCENOTALLOWED 0x80660000 // The reference␣
↪type between the nodes is already defined.
#define UA_STATUSCODE_BADINVALIDSELFREFERENCE 0x80670000 // The server does not␣
↪allow this type of self reference on this node.
#define UA_STATUSCODE_BADREFERENCELOCALONLY 0x80680000 // The reference type is␣
↪not valid for a reference to a remote server.
#define UA_STATUSCODE_BADNODELETERIGHTS 0x80690000 // The server will not allow␣
↪the node to be deleted.
#define UA_STATUSCODE_UNCERTAINREFERENCENOTDELETED 0x40bc0000 // The server was␣
↪not able to delete all target references.
#define UA_STATUSCODE_BADSERVERINDEXINVALID 0x806a0000 // The server index is not␣
↪valid.
#define UA_STATUSCODE_BADVIEWIDUNKNOWN 0x806b0000 // The view id does not refer to␣
↪a valid view node.
#define UA_STATUSCODE_BADVIEWTIMESTAMPINVALID 0x80c90000 // The view timestamp is␣
↪not available or not supported.
#define UA_STATUSCODE_BADVIEWPARAMETERMISMATCH 0x80ca0000 // The view parameters␣
↪are not consistent with each other.
#define UA_STATUSCODE_BADVIEWVERSIONINVALID 0x80cb0000 // The view version is not␣
↪available or not supported.
#define UA_STATUSCODE_UNCERTAINNOTALLNODESAVAILABLE 0x40c00000 // The list of␣
↪references may not be complete because the underlying system is not available.
#define UA_STATUSCODE_GOODRESULTSMAYBEINCOMPLETE 0x00ba0000 // The server should␣
↪have followed a reference to a node in a remote server but did not. The result␣
↪set may be incomplete.
```

```
#define UA_STATUSCODE_BADNOTTYPEDEFINITION 0x80c80000 // The provided Nodeid was␣
→not a type definition nodeid.
#define UA_STATUSCODE_UNCERTAINREFERENCEOUTOFSERVER 0x406c0000 // One of the␣
→references to follow in the relative path references to a node in the address␣
→space in another server.
#define UA_STATUSCODE_BADTOOMANYMATCHES 0x806d0000 // The requested operation has␣
→too many matches to return.
#define UA_STATUSCODE_BADQUERYTOOCOMPLEX 0x806e0000 // The requested operation␣
→requires too many resources in the server.
#define UA_STATUSCODE_BADNOMATCH 0x806f0000 // The requested operation has no␣
→match to return.
#define UA_STATUSCODE_BADMAXAGEINVALID 0x80700000 // The max age parameter is␣
→invalid.
#define UA_STATUSCODE_BADSECURITYMODEINSUFFICIENT 0x80e60000 // The operation is␣
→not permitted over the current secure channel.
#define UA_STATUSCODE_BADHISTORYOPERATIONINVALID 0x80710000 // The history details␣
→parameter is not valid.
#define UA_STATUSCODE_BADHISTORYOPERATIONUNSUPPORTED 0x80720000 // The server does␣
→not support the requested operation.
#define UA_STATUSCODE_BADINVALIDTIMESTAMPARGUMENT 0x80bd0000 // The defined␣
→timestamp to return was invalid.
#define UA_STATUSCODE_BADWRITENOTSUPPORTED 0x80730000 // The server not does␣
→support writing the combination of value, status and timestamps provided.
#define UA_STATUSCODE_BADTYPEMISMATCH 0x80740000 // The value supplied for the␣
→attribute is not of the same type as the attribute's value.
#define UA_STATUSCODE_BADMETHODINVALID 0x80750000 // The method id does not refer␣
→to a method for the specified object.
#define UA_STATUSCODE_BADARGUMENTSMISSING 0x80760000 // The client did not specify␣
→all of the input arguments for the method.
#define UA_STATUSCODE_BADTOOMANYSUBSCRIPTIONS 0x80770000 // The server has reached␣
→its  maximum number of subscriptions.
#define UA_STATUSCODE_BADTOOMANYPUBLISHREQUESTS 0x80780000 // The server has␣
→reached the maximum number of queued publish requests.
#define UA_STATUSCODE_BADNOSUBSCRIPTION 0x80790000 // There is no subscription␣
→available for this session.
#define UA_STATUSCODE_BADSEQUENCENUMBERUNKNOWN 0x807a0000 // The sequence number␣
→is unknown to the server.
#define UA_STATUSCODE_BADMESSAGENOTAVAILABLE 0x807b0000 // The requested␣
→notification message is no longer available.
#define UA_STATUSCODE_BADINSUFFICIENTCLIENTPROFILE 0x807c0000 // The Client of the␣
→current Session does not support one or more Profiles that are necessary for the␣
→Subscription.
#define UA_STATUSCODE_BADSTATENOTACTIVE 0x80bf0000 // The sub-state machine is not␣
→currently active.
#define UA_STATUSCODE_BADTCPSERVERTOOBUSY 0x807d0000 // The server cannot process␣
→the request because it is too busy.
#define UA_STATUSCODE_BADTCPMESSAGETYPEINVALID 0x807e0000 // The type of the␣
→message specified in the header invalid.
#define UA_STATUSCODE_BADTCPSECURECHANNELUNKNOWN 0x807f0000 // The SecureChannelId␣
→and/or TokenId are not currently in use.
#define UA_STATUSCODE_BADTCPMESSAGETOOLARGE 0x80800000 // The size of the message␣
→specified in the header is too large.
#define UA_STATUSCODE_BADTCPNOTENOUGHRESOURCES 0x80810000 // There are not enough␣
→resources to process the request.
#define UA_STATUSCODE_BADTCPINTERNALERROR 0x80820000 // An internal error occurred.
#define UA_STATUSCODE_BADTCPENDPOINTURLINVALID 0x80830000 // The Server does not␣
→recognize the QueryString specified.
#define UA_STATUSCODE_BADREQUESTINTERRUPTED 0x80840000 // The request could not be␣
→sent because of a network interruption.
#define UA_STATUSCODE_BADREQUESTTIMEOUT 0x80850000 // Timeout occurred while␣
→processing the request.
```

```
#define UA_STATUSCODE_BADSECURECHANNELCLOSED 0x80860000 // The secure channel has␣
↪been closed.
#define UA_STATUSCODE_BADSECURECHANNELTOKENUNKNOWN 0x80870000 // The token has␣
↪expired or is not recognized.
#define UA_STATUSCODE_BADSEQUENCENUMBERINVALID 0x80880000 // The sequence number␣
↪is not valid.
#define UA_STATUSCODE_BADPROTOCOLVERSIONUNSUPPORTED 0x80be0000 // The applications␣
↪do not have compatible protocol versions.
#define UA_STATUSCODE_BADCONFIGURATIONERROR 0x80890000 // There is a problem with␣
↪the configuration that affects the usefulness of the value.
#define UA_STATUSCODE_BADNOTCONNECTED 0x808a0000 // The variable should receive␣
↪its value from another variable, but has never been configured to do so.
#define UA_STATUSCODE_BADDEVICEFAILURE 0x808b0000 // There has been a failure in␣
↪the device/data source that generates the value that has affected the value.
#define UA_STATUSCODE_BADSENSORFAILURE 0x808c0000 // There has been a failure in␣
↪the sensor from which the value is derived by the device/data source.
#define UA_STATUSCODE_BADOUTOFSERVICE 0x808d0000 // The source of the data is not␣
↪operational.
#define UA_STATUSCODE_BADDEADBANDFILTERINVALID 0x808e0000 // The deadband filter␣
↪is not valid.
#define UA_STATUSCODE_UNCERTAINNOCOMMUNICATIONLASTUSABLEVALUE 0x408f0000 //␣
↪Communication to the data source has failed. The variable value is the last␣
↪value that had a good quality.
#define UA_STATUSCODE_UNCERTAINLASTUSABLEVALUE 0x40900000 // Whatever was updating␣
↪this value has stopped doing so.
#define UA_STATUSCODE_UNCERTAINSUBSTITUTEVALUE 0x40910000 // The value is an␣
↪operational value that was manually overwritten.
#define UA_STATUSCODE_UNCERTAININITIALVALUE 0x40920000 // The value is an initial␣
↪value for a variable that normally receives its value from another variable.
#define UA_STATUSCODE_UNCERTAINSENSORNOTACCURATE 0x40930000 // The value is at one␣
↪of the sensor limits.
#define UA_STATUSCODE_UNCERTAINENGINEERINGUNITSEXCEEDED 0x40940000 // The value is␣
↪outside of the range of values defined for this parameter.
#define UA_STATUSCODE_UNCERTAINSUBNORMAL 0x40950000 // The value is derived from␣
↪multiple sources and has less than the required number of Good sources.
#define UA_STATUSCODE_GOODLOCALOVERRIDE 0x00960000 // The value has been␣
↪overridden.
#define UA_STATUSCODE_BADREFRESHINPROGRESS 0x80970000 // This Condition refresh␣
↪failed, a Condition refresh operation is already in progress.
#define UA_STATUSCODE_BADCONDITIONALREADYDISABLED 0x80980000 // This condition has␣
↪already been disabled.
#define UA_STATUSCODE_BADCONDITIONALREADYENABLED 0x80cc0000 // This condition has␣
↪already been enabled.
#define UA_STATUSCODE_BADCONDITIONDISABLED 0x80990000 // Property not available,␣
↪this condition is disabled.
#define UA_STATUSCODE_BADEVENTIDUNKNOWN 0x809a0000 // The specified event id is␣
↪not recognized.
#define UA_STATUSCODE_BADEVENTNOTACKNOWLEDGEABLE 0x80bb0000 // The event cannot be␣
↪acknowledged.
#define UA_STATUSCODE_BADDIALOGNOTACTIVE 0x80cd0000 // The dialog condition is not␣
↪active.
#define UA_STATUSCODE_BADDIALOGRESPONSEINVALID 0x80ce0000 // The response is not␣
↪valid for the dialog.
#define UA_STATUSCODE_BADCONDITIONBRANCHALREADYACKED 0x80cf0000 // The condition␣
↪branch has already been acknowledged.
#define UA_STATUSCODE_BADCONDITIONBRANCHALREADYCONFIRMED 0x80d00000 // The␣
↪condition branch has already been confirmed.
#define UA_STATUSCODE_BADCONDITIONALREADYSHELVED 0x80d10000 // The condition has␣
↪already been shelved.
#define UA_STATUSCODE_BADCONDITIONNOTSHELVED 0x80d20000 // The condition is not␣
↪currently shelved.
```

```
#define UA_STATUSCODE_BADSHELVINGTIMEOUTOFRANGE 0x80d30000 // The shelving time␣
→not within an acceptable range.
#define UA_STATUSCODE_BADNODATA 0x809b0000 // No data exists for the requested␣
→time range or event filter.
#define UA_STATUSCODE_BADBOUNDNOTFOUND 0x80d70000 // No data found to provide␣
→upper or lower bound value.
#define UA_STATUSCODE_BADBOUNDNOTSUPPORTED 0x80d80000 // The server cannot␣
→retrieve a bound for the variable.
#define UA_STATUSCODE_BADDATALOST 0x809d0000 // Data is missing due to collection␣
→started/stopped/lost.
#define UA_STATUSCODE_BADDATAUNAVAILABLE 0x809e0000 // Expected data is␣
→unavailable for the requested time range due to an un-mounted volume, an off-
→line archive or tape, or similar reason for temporary unavailability.
#define UA_STATUSCODE_BADENTRYEXISTS 0x809f0000 // The data or event was not␣
→successfully inserted because a matching entry exists.
#define UA_STATUSCODE_BADNOENTRYEXISTS 0x80a00000 // The data or event was not␣
→successfully updated because no matching entry exists.
#define UA_STATUSCODE_BADTIMESTAMPNOTSUPPORTED 0x80a10000 // The client requested␣
→history using a timestamp format the server does not support (i.e requested␣
→ServerTimestamp when server only supports SourceTimestamp).
#define UA_STATUSCODE_GOODENTRYINSERTED 0x00a20000 // The data or event was␣
→successfully inserted into the historical database.
#define UA_STATUSCODE_GOODENTRYREPLACED 0x00a30000 // The data or event field was␣
→successfully replaced in the historical database.
#define UA_STATUSCODE_UNCERTAINDATASUBNORMAL 0x40a40000 // The value is derived␣
→from multiple values and has less than the required number of Good values.
#define UA_STATUSCODE_GOODNODATA 0x00a50000 // No data exists for the requested␣
→time range or event filter.
#define UA_STATUSCODE_GOODMOREDATA 0x00a60000 // The data or event field was␣
→successfully replaced in the historical database.
#define UA_STATUSCODE_BADAGGREGATELISTMISMATCH 0x80d40000 // The requested number␣
→of Aggregates does not match the requested number of NodeIds.
#define UA_STATUSCODE_BADAGGREGATENOTSUPPORTED 0x80d50000 // The requested␣
→Aggregate is not support by the server.
#define UA_STATUSCODE_BADAGGREGATEINVALIDINPUTS 0x80d60000 // The aggregate value␣
→could not be derived due to invalid data inputs.
#define UA_STATUSCODE_BADAGGREGATECONFIGURATIONREJECTED 0x80da0000 // The␣
→aggregate configuration is not valid for specified node.
#define UA_STATUSCODE_GOODDATAIGNORED 0x00d90000 // The request pecifies fields␣
→which are not valid for the EventType or cannot be saved by the historian.
#define UA_STATUSCODE_BADREQUESTNOTALLOWED 0x80e40000 // The request was rejected␣
→by the server because it did not meet the criteria set by the server.
#define UA_STATUSCODE_GOODEDITED 0x00dc0000 // The value does not come from the␣
→real source and has been edited by the server.
#define UA_STATUSCODE_GOODPOSTACTIONFAILED 0x00dd0000 // There was an error in␣
→execution of these post-actions.
#define UA_STATUSCODE_UNCERTAINDOMINANTVALUECHANGED 0x40de0000 // The related␣
→EngineeringUnit has been changed but the Variable Value is still provided based␣
→on the previous unit.
#define UA_STATUSCODE_GOODDEPENDENTVALUECHANGED 0x00e00000 // A dependent value␣
→has been changed but the change has not been applied to the device.
#define UA_STATUSCODE_BADDOMINANTVALUECHANGED 0x80e10000 // The related␣
→EngineeringUnit has been changed but this change has not been applied to the␣
→device. The Variable Value is still dependent on the previous unit but its␣
→status is currently Bad.
#define UA_STATUSCODE_UNCERTAINDEPENDENTVALUECHANGED 0x40e20000 // A dependent␣
→value has been changed but the change has not been applied to the device. The␣
→quality of the dominant variable is uncertain.
#define UA_STATUSCODE_BADDEPENDENTVALUECHANGED 0x80e30000 // A dependent value has␣
→been changed but the change has not been applied to the device. The quality of␣
→the dominant variable is Bad.
```

```
#define UA_STATUSCODE_GOODCOMMUNICATIONEVENT 0x00a70000 // The communication layer␣
↪has raised an event.
#define UA_STATUSCODE_GOODSHUTDOWNEVENT 0x00a80000 // The system is shutting down.
#define UA_STATUSCODE_GOODCALLAGAIN 0x00a90000 // The operation is not finished␣
↪and needs to be called again.
#define UA_STATUSCODE_GOODNONCRITICALTIMEOUT 0x00aa0000 // A non-critical timeout␣
↪occurred.
#define UA_STATUSCODE_BADINVALIDARGUMENT 0x80ab0000 // One or more arguments are␣
↪invalid.
#define UA_STATUSCODE_BADCONNECTIONREJECTED 0x80ac0000 // Could not establish a␣
↪network connection to remote server.
#define UA_STATUSCODE_BADDISCONNECT 0x80ad0000 // The server has disconnected from␣
↪the client.
#define UA_STATUSCODE_BADCONNECTIONCLOSED 0x80ae0000 // The network connection has␣
↪been closed.
#define UA_STATUSCODE_BADINVALIDSTATE 0x80af0000 // The operation cannot be␣
↪completed because the object is closed, uninitialized or in some other invalid␣
↪state.
#define UA_STATUSCODE_BADENDOFSTREAM 0x80b00000 // Cannot move beyond end of the␣
↪stream.
#define UA_STATUSCODE_BADNODATAAVAILABLE 0x80b10000 // No data is currently␣
↪available for reading from a non-blocking stream.
#define UA_STATUSCODE_BADWAITINGFORRESPONSE 0x80b20000 // The asynchronous␣
↪operation is waiting for a response.
#define UA_STATUSCODE_BADOPERATIONABANDONED 0x80b30000 // The asynchronous␣
↪operation was abandoned by the caller.
#define UA_STATUSCODE_BADEXPECTEDSTREAMTOBLOCK 0x80b40000 // The stream did not␣
↪return all data requested (possibly because it is a non-blocking stream).
#define UA_STATUSCODE_BADWOULDBLOCK 0x80b50000 // Non blocking behaviour is␣
↪required and the operation would block.
#define UA_STATUSCODE_BADSYNTAXERROR 0x80b60000 // A value had an invalid syntax.
#define UA_STATUSCODE_BADMAXCONNECTIONSREACHED 0x80b70000 // The operation could␣
↪not be finished because all available connections are in use.

/* These StatusCodes are manually generated. */
#define UA_STATUSCODE_INFOTYPE_DATAVALUE 0x00000400
#define UA_STATUSCODE_INFOBITS_OVERFLOW 0x00000080
```

10.5 Namespace Zero NodeIds

Numeric identifiers of standard-defined nodes in namespace zero. The following definitions are autogenerated from the `NodeIds.csv` file provided with the OPC UA standard.

```
#define UA_NS0ID_BOOLEAN 1 // DataType
#define UA_NS0ID_SBYTE 2 // DataType
#define UA_NS0ID_BYTE 3 // DataType
#define UA_NS0ID_INT16 4 // DataType
#define UA_NS0ID_UINT16 5 // DataType
#define UA_NS0ID_INT32 6 // DataType
#define UA_NS0ID_UINT32 7 // DataType
#define UA_NS0ID_INT64 8 // DataType
#define UA_NS0ID_UINT64 9 // DataType
#define UA_NS0ID_FLOAT 10 // DataType
#define UA_NS0ID_DOUBLE 11 // DataType
#define UA_NS0ID_STRING 12 // DataType
#define UA_NS0ID_DATETIME 13 // DataType
#define UA_NS0ID_GUID 14 // DataType
#define UA_NS0ID_BYTESTRING 15 // DataType
#define UA_NS0ID_XMLELEMENT 16 // DataType
```

```
#define UA_NS0ID_NODEID 17 // DataType
#define UA_NS0ID_EXPANDEDNODEID 18 // DataType
#define UA_NS0ID_STATUSCODE 19 // DataType
#define UA_NS0ID_QUALIFIEDNAME 20 // DataType
#define UA_NS0ID_LOCALIZEDTEXT 21 // DataType
#define UA_NS0ID_STRUCTURE 22 // DataType
#define UA_NS0ID_DATAVALUE 23 // DataType
#define UA_NS0ID_BASEDATATYPE 24 // DataType
#define UA_NS0ID_DIAGNOSTICINFO 25 // DataType
#define UA_NS0ID_NUMBER 26 // DataType
#define UA_NS0ID_INTEGER 27 // DataType
#define UA_NS0ID_UINTEGER 28 // DataType
#define UA_NS0ID_ENUMERATION 29 // DataType
#define UA_NS0ID_IMAGE 30 // DataType
#define UA_NS0ID_REFERENCES 31 // ReferenceType
#define UA_NS0ID_NONHIERARCHICALREFERENCES 32 // ReferenceType
#define UA_NS0ID_HIERARCHICALREFERENCES 33 // ReferenceType
#define UA_NS0ID_HASCHILD 34 // ReferenceType
#define UA_NS0ID_ORGANIZES 35 // ReferenceType
#define UA_NS0ID_HASEVENTSOURCE 36 // ReferenceType
#define UA_NS0ID_HASMODELLINGRULE 37 // ReferenceType
#define UA_NS0ID_HASENCODING 38 // ReferenceType
#define UA_NS0ID_HASDESCRIPTION 39 // ReferenceType
#define UA_NS0ID_HASTYPEDEFINITION 40 // ReferenceType
#define UA_NS0ID_GENERATESEVENT 41 // ReferenceType
#define UA_NS0ID_AGGREGATES 44 // ReferenceType
#define UA_NS0ID_HASSUBTYPE 45 // ReferenceType
#define UA_NS0ID_HASPROPERTY 46 // ReferenceType
#define UA_NS0ID_HASCOMPONENT 47 // ReferenceType
#define UA_NS0ID_HASNOTIFIER 48 // ReferenceType
#define UA_NS0ID_HASORDEREDCOMPONENT 49 // ReferenceType
#define UA_NS0ID_FROMSTATE 51 // ReferenceType
#define UA_NS0ID_TOSTATE 52 // ReferenceType
#define UA_NS0ID_HASCAUSE 53 // ReferenceType
#define UA_NS0ID_HASEFFECT 54 // ReferenceType
#define UA_NS0ID_HASHISTORICALCONFIGURATION 56 // ReferenceType
#define UA_NS0ID_BASEOBJECTTYPE 58 // ObjectType
#define UA_NS0ID_FOLDERTYPE 61 // ObjectType
#define UA_NS0ID_BASEVARIABLETYPE 62 // VariableType
#define UA_NS0ID_BASEDATAVARIABLETYPE 63 // VariableType
#define UA_NS0ID_PROPERTYTYPE 68 // VariableType
#define UA_NS0ID_DATATYPEDESCRIPTIONTYPE 69 // VariableType
#define UA_NS0ID_DATATYPEDICTIONARYTYPE 72 // VariableType
#define UA_NS0ID_DATATYPESYSTEMTYPE 75 // ObjectType
#define UA_NS0ID_DATATYPEENCODINGTYPE 76 // ObjectType
#define UA_NS0ID_MODELLINGRULETYPE 77 // ObjectType
#define UA_NS0ID_MODELLINGRULE_MANDATORY 78 // Object
#define UA_NS0ID_MODELLINGRULE_MANDATORYSHARED 79 // Object
#define UA_NS0ID_MODELLINGRULE_OPTIONAL 80 // Object
#define UA_NS0ID_MODELLINGRULE_EXPOSESITSARRAY 83 // Object
#define UA_NS0ID_ROOTFOLDER 84 // Object
#define UA_NS0ID_OBJECTSFOLDER 85 // Object
#define UA_NS0ID_TYPESFOLDER 86 // Object
#define UA_NS0ID_VIEWSFOLDER 87 // Object
#define UA_NS0ID_OBJECTTYPESFOLDER 88 // Object
#define UA_NS0ID_VARIABLETYPESFOLDER 89 // Object
#define UA_NS0ID_DATATYPESFOLDER 90 // Object
#define UA_NS0ID_REFERENCETYPESFOLDER 91 // Object
#define UA_NS0ID_XMLSCHEMA_TYPESYSTEM 92 // Object
#define UA_NS0ID_OPCBINARYSCHEMA_TYPESYSTEM 93 // Object
#define UA_NS0ID_MODELLINGRULE_MANDATORY_NAMINGRULE 112 // Variable
```

```
#define UA_NS0ID_MODELLINGRULE_OPTIONAL_NAMINGRULE 113 // Variable
#define UA_NS0ID_MODELLINGRULE_EXPOSESITSARRAY_NAMINGRULE 114 // Variable
#define UA_NS0ID_MODELLINGRULE_MANDATORYSHARED_NAMINGRULE 116 // Variable
#define UA_NS0ID_HASSUBSTATEMACHINE 117 // ReferenceType
#define UA_NS0ID_NAMINGRULETYPE 120 // DataType
#define UA_NS0ID_DECIMAL128 121 // DataType
#define UA_NS0ID_IDTYPE 256 // DataType
#define UA_NS0ID_NODECLASS 257 // DataType
#define UA_NS0ID_NODE 258 // DataType
#define UA_NS0ID_NODE_ENCODING_DEFAULTXML 259 // Object
#define UA_NS0ID_NODE_ENCODING_DEFAULTBINARY 260 // Object
#define UA_NS0ID_OBJECTNODE 261 // DataType
#define UA_NS0ID_OBJECTNODE_ENCODING_DEFAULTXML 262 // Object
#define UA_NS0ID_OBJECTNODE_ENCODING_DEFAULTBINARY 263 // Object
#define UA_NS0ID_OBJECTTYPENODE 264 // DataType
#define UA_NS0ID_OBJECTTYPENODE_ENCODING_DEFAULTXML 265 // Object
#define UA_NS0ID_OBJECTTYPENODE_ENCODING_DEFAULTBINARY 266 // Object
#define UA_NS0ID_VARIABLENODE 267 // DataType
#define UA_NS0ID_VARIABLENODE_ENCODING_DEFAULTXML 268 // Object
#define UA_NS0ID_VARIABLENODE_ENCODING_DEFAULTBINARY 269 // Object
#define UA_NS0ID_VARIABLETYPENODE 270 // DataType
#define UA_NS0ID_VARIABLETYPENODE_ENCODING_DEFAULTXML 271 // Object
#define UA_NS0ID_VARIABLETYPENODE_ENCODING_DEFAULTBINARY 272 // Object
#define UA_NS0ID_REFERENCETYPENODE 273 // DataType
#define UA_NS0ID_REFERENCETYPENODE_ENCODING_DEFAULTXML 274 // Object
#define UA_NS0ID_REFERENCETYPENODE_ENCODING_DEFAULTBINARY 275 // Object
#define UA_NS0ID_METHODNODE 276 // DataType
#define UA_NS0ID_METHODNODE_ENCODING_DEFAULTXML 277 // Object
#define UA_NS0ID_METHODNODE_ENCODING_DEFAULTBINARY 278 // Object
#define UA_NS0ID_VIEWNODE 279 // DataType
#define UA_NS0ID_VIEWNODE_ENCODING_DEFAULTXML 280 // Object
#define UA_NS0ID_VIEWNODE_ENCODING_DEFAULTBINARY 281 // Object
#define UA_NS0ID_DATATYPENODE 282 // DataType
#define UA_NS0ID_DATATYPENODE_ENCODING_DEFAULTXML 283 // Object
#define UA_NS0ID_DATATYPENODE_ENCODING_DEFAULTBINARY 284 // Object
#define UA_NS0ID_REFERENCENODE 285 // DataType
#define UA_NS0ID_REFERENCENODE_ENCODING_DEFAULTXML 286 // Object
#define UA_NS0ID_REFERENCENODE_ENCODING_DEFAULTBINARY 287 // Object
#define UA_NS0ID_INTEGERID 288 // DataType
#define UA_NS0ID_COUNTER 289 // DataType
#define UA_NS0ID_DURATION 290 // DataType
#define UA_NS0ID_NUMERICRANGE 291 // DataType
#define UA_NS0ID_TIME 292 // DataType
#define UA_NS0ID_DATE 293 // DataType
#define UA_NS0ID_UTCTIME 294 // DataType
#define UA_NS0ID_LOCALEID 295 // DataType
#define UA_NS0ID_ARGUMENT 296 // DataType
#define UA_NS0ID_ARGUMENT_ENCODING_DEFAULTXML 297 // Object
#define UA_NS0ID_ARGUMENT_ENCODING_DEFAULTBINARY 298 // Object
#define UA_NS0ID_STATUSRESULT 299 // DataType
#define UA_NS0ID_STATUSRESULT_ENCODING_DEFAULTXML 300 // Object
#define UA_NS0ID_STATUSRESULT_ENCODING_DEFAULTBINARY 301 // Object
#define UA_NS0ID_MESSAGESECURITYMODE 302 // DataType
#define UA_NS0ID_USERTOKENTYPE 303 // DataType
#define UA_NS0ID_USERTOKENPOLICY 304 // DataType
#define UA_NS0ID_USERTOKENPOLICY_ENCODING_DEFAULTXML 305 // Object
#define UA_NS0ID_USERTOKENPOLICY_ENCODING_DEFAULTBINARY 306 // Object
#define UA_NS0ID_APPLICATIONTYPE 307 // DataType
#define UA_NS0ID_APPLICATIONDESCRIPTION 308 // DataType
#define UA_NS0ID_APPLICATIONDESCRIPTION_ENCODING_DEFAULTXML 309 // Object
#define UA_NS0ID_APPLICATIONDESCRIPTION_ENCODING_DEFAULTBINARY 310 // Object
```

```
#define UA_NS0ID_APPLICATIONINSTANCECERTIFICATE 311 // DataType
#define UA_NS0ID_ENDPOINTDESCRIPTION 312 // DataType
#define UA_NS0ID_ENDPOINTDESCRIPTION_ENCODING_DEFAULTXML 313 // Object
#define UA_NS0ID_ENDPOINTDESCRIPTION_ENCODING_DEFAULTBINARY 314 // Object
#define UA_NS0ID_SECURITYTOKENREQUESTTYPE 315 // DataType
#define UA_NS0ID_USERIDENTITYTOKEN 316 // DataType
#define UA_NS0ID_USERIDENTITYTOKEN_ENCODING_DEFAULTXML 317 // Object
#define UA_NS0ID_USERIDENTITYTOKEN_ENCODING_DEFAULTBINARY 318 // Object
#define UA_NS0ID_ANONYMOUSIDENTITYTOKEN 319 // DataType
#define UA_NS0ID_ANONYMOUSIDENTITYTOKEN_ENCODING_DEFAULTXML 320 // Object
#define UA_NS0ID_ANONYMOUSIDENTITYTOKEN_ENCODING_DEFAULTBINARY 321 // Object
#define UA_NS0ID_USERNAMEIDENTITYTOKEN 322 // DataType
#define UA_NS0ID_USERNAMEIDENTITYTOKEN_ENCODING_DEFAULTXML 323 // Object
#define UA_NS0ID_USERNAMEIDENTITYTOKEN_ENCODING_DEFAULTBINARY 324 // Object
#define UA_NS0ID_X509IDENTITYTOKEN 325 // DataType
#define UA_NS0ID_X509IDENTITYTOKEN_ENCODING_DEFAULTXML 326 // Object
#define UA_NS0ID_X509IDENTITYTOKEN_ENCODING_DEFAULTBINARY 327 // Object
#define UA_NS0ID_ENDPOINTCONFIGURATION 331 // DataType
#define UA_NS0ID_ENDPOINTCONFIGURATION_ENCODING_DEFAULTXML 332 // Object
#define UA_NS0ID_ENDPOINTCONFIGURATION_ENCODING_DEFAULTBINARY 333 // Object
#define UA_NS0ID_BUILDINFO 338 // DataType
#define UA_NS0ID_BUILDINFO_ENCODING_DEFAULTXML 339 // Object
#define UA_NS0ID_BUILDINFO_ENCODING_DEFAULTBINARY 340 // Object
#define UA_NS0ID_SIGNEDSOFTWARECERTIFICATE 344 // DataType
#define UA_NS0ID_SIGNEDSOFTWARECERTIFICATE_ENCODING_DEFAULTXML 345 // Object
#define UA_NS0ID_SIGNEDSOFTWARECERTIFICATE_ENCODING_DEFAULTBINARY 346 // Object
#define UA_NS0ID_ATTRIBUTEWRITEMASK 347 // DataType
#define UA_NS0ID_NODEATTRIBUTESMASK 348 // DataType
#define UA_NS0ID_NODEATTRIBUTES 349 // DataType
#define UA_NS0ID_NODEATTRIBUTES_ENCODING_DEFAULTXML 350 // Object
#define UA_NS0ID_NODEATTRIBUTES_ENCODING_DEFAULTBINARY 351 // Object
#define UA_NS0ID_OBJECTATTRIBUTES 352 // DataType
#define UA_NS0ID_OBJECTATTRIBUTES_ENCODING_DEFAULTXML 353 // Object
#define UA_NS0ID_OBJECTATTRIBUTES_ENCODING_DEFAULTBINARY 354 // Object
#define UA_NS0ID_VARIABLEATTRIBUTES 355 // DataType
#define UA_NS0ID_VARIABLEATTRIBUTES_ENCODING_DEFAULTXML 356 // Object
#define UA_NS0ID_VARIABLEATTRIBUTES_ENCODING_DEFAULTBINARY 357 // Object
#define UA_NS0ID_METHODATTRIBUTES 358 // DataType
#define UA_NS0ID_METHODATTRIBUTES_ENCODING_DEFAULTXML 359 // Object
#define UA_NS0ID_METHODATTRIBUTES_ENCODING_DEFAULTBINARY 360 // Object
#define UA_NS0ID_OBJECTTYPEATTRIBUTES 361 // DataType
#define UA_NS0ID_OBJECTTYPEATTRIBUTES_ENCODING_DEFAULTXML 362 // Object
#define UA_NS0ID_OBJECTTYPEATTRIBUTES_ENCODING_DEFAULTBINARY 363 // Object
#define UA_NS0ID_VARIABLETYPEATTRIBUTES 364 // DataType
#define UA_NS0ID_VARIABLETYPEATTRIBUTES_ENCODING_DEFAULTXML 365 // Object
#define UA_NS0ID_VARIABLETYPEATTRIBUTES_ENCODING_DEFAULTBINARY 366 // Object
#define UA_NS0ID_REFERENCETYPEATTRIBUTES 367 // DataType
#define UA_NS0ID_REFERENCETYPEATTRIBUTES_ENCODING_DEFAULTXML 368 // Object
#define UA_NS0ID_REFERENCETYPEATTRIBUTES_ENCODING_DEFAULTBINARY 369 // Object
#define UA_NS0ID_DATATYPEATTRIBUTES 370 // DataType
#define UA_NS0ID_DATATYPEATTRIBUTES_ENCODING_DEFAULTXML 371 // Object
#define UA_NS0ID_DATATYPEATTRIBUTES_ENCODING_DEFAULTBINARY 372 // Object
#define UA_NS0ID_VIEWATTRIBUTES 373 // DataType
#define UA_NS0ID_VIEWATTRIBUTES_ENCODING_DEFAULTXML 374 // Object
#define UA_NS0ID_VIEWATTRIBUTES_ENCODING_DEFAULTBINARY 375 // Object
#define UA_NS0ID_ADDNODESITEM 376 // DataType
#define UA_NS0ID_ADDNODESITEM_ENCODING_DEFAULTXML 377 // Object
#define UA_NS0ID_ADDNODESITEM_ENCODING_DEFAULTBINARY 378 // Object
#define UA_NS0ID_ADDREFERENCESITEM 379 // DataType
#define UA_NS0ID_ADDREFERENCESITEM_ENCODING_DEFAULTXML 380 // Object
#define UA_NS0ID_ADDREFERENCESITEM_ENCODING_DEFAULTBINARY 381 // Object
```

```
#define UA_NS0ID_DELETENODESITEM 382 // DataType
#define UA_NS0ID_DELETENODESITEM_ENCODING_DEFAULTXML 383 // Object
#define UA_NS0ID_DELETENODESITEM_ENCODING_DEFAULTBINARY 384 // Object
#define UA_NS0ID_DELETEREFERENCESITEM 385 // DataType
#define UA_NS0ID_DELETEREFERENCESITEM_ENCODING_DEFAULTXML 386 // Object
#define UA_NS0ID_DELETEREFERENCESITEM_ENCODING_DEFAULTBINARY 387 // Object
#define UA_NS0ID_SESSIONAUTHENTICATIONTOKEN 388 // DataType
#define UA_NS0ID_REQUESTHEADER 389 // DataType
#define UA_NS0ID_REQUESTHEADER_ENCODING_DEFAULTXML 390 // Object
#define UA_NS0ID_REQUESTHEADER_ENCODING_DEFAULTBINARY 391 // Object
#define UA_NS0ID_RESPONSEHEADER 392 // DataType
#define UA_NS0ID_RESPONSEHEADER_ENCODING_DEFAULTXML 393 // Object
#define UA_NS0ID_RESPONSEHEADER_ENCODING_DEFAULTBINARY 394 // Object
#define UA_NS0ID_SERVICEFAULT 395 // DataType
#define UA_NS0ID_SERVICEFAULT_ENCODING_DEFAULTXML 396 // Object
#define UA_NS0ID_SERVICEFAULT_ENCODING_DEFAULTBINARY 397 // Object
#define UA_NS0ID_FINDSERVERSREQUEST 420 // DataType
#define UA_NS0ID_FINDSERVERSREQUEST_ENCODING_DEFAULTXML 421 // Object
#define UA_NS0ID_FINDSERVERSREQUEST_ENCODING_DEFAULTBINARY 422 // Object
#define UA_NS0ID_FINDSERVERSRESPONSE 423 // DataType
#define UA_NS0ID_FINDSERVERSRESPONSE_ENCODING_DEFAULTXML 424 // Object
#define UA_NS0ID_FINDSERVERSRESPONSE_ENCODING_DEFAULTBINARY 425 // Object
#define UA_NS0ID_GETENDPOINTSREQUEST 426 // DataType
#define UA_NS0ID_GETENDPOINTSREQUEST_ENCODING_DEFAULTXML 427 // Object
#define UA_NS0ID_GETENDPOINTSREQUEST_ENCODING_DEFAULTBINARY 428 // Object
#define UA_NS0ID_GETENDPOINTSRESPONSE 429 // DataType
#define UA_NS0ID_GETENDPOINTSRESPONSE_ENCODING_DEFAULTXML 430 // Object
#define UA_NS0ID_GETENDPOINTSRESPONSE_ENCODING_DEFAULTBINARY 431 // Object
#define UA_NS0ID_REGISTEREDSERVER 432 // DataType
#define UA_NS0ID_REGISTEREDSERVER_ENCODING_DEFAULTXML 433 // Object
#define UA_NS0ID_REGISTEREDSERVER_ENCODING_DEFAULTBINARY 434 // Object
#define UA_NS0ID_REGISTERSERVERREQUEST 435 // DataType
#define UA_NS0ID_REGISTERSERVERREQUEST_ENCODING_DEFAULTXML 436 // Object
#define UA_NS0ID_REGISTERSERVERREQUEST_ENCODING_DEFAULTBINARY 437 // Object
#define UA_NS0ID_REGISTERSERVERRESPONSE 438 // DataType
#define UA_NS0ID_REGISTERSERVERRESPONSE_ENCODING_DEFAULTXML 439 // Object
#define UA_NS0ID_REGISTERSERVERRESPONSE_ENCODING_DEFAULTBINARY 440 // Object
#define UA_NS0ID_CHANNELSECURITYTOKEN 441 // DataType
#define UA_NS0ID_CHANNELSECURITYTOKEN_ENCODING_DEFAULTXML 442 // Object
#define UA_NS0ID_CHANNELSECURITYTOKEN_ENCODING_DEFAULTBINARY 443 // Object
#define UA_NS0ID_OPENSECURECHANNELREQUEST 444 // DataType
#define UA_NS0ID_OPENSECURECHANNELREQUEST_ENCODING_DEFAULTXML 445 // Object
#define UA_NS0ID_OPENSECURECHANNELREQUEST_ENCODING_DEFAULTBINARY 446 // Object
#define UA_NS0ID_OPENSECURECHANNELRESPONSE 447 // DataType
#define UA_NS0ID_OPENSECURECHANNELRESPONSE_ENCODING_DEFAULTXML 448 // Object
#define UA_NS0ID_OPENSECURECHANNELRESPONSE_ENCODING_DEFAULTBINARY 449 // Object
#define UA_NS0ID_CLOSESECURECHANNELREQUEST 450 // DataType
#define UA_NS0ID_CLOSESECURECHANNELREQUEST_ENCODING_DEFAULTXML 451 // Object
#define UA_NS0ID_CLOSESECURECHANNELREQUEST_ENCODING_DEFAULTBINARY 452 // Object
#define UA_NS0ID_CLOSESECURECHANNELRESPONSE 453 // DataType
#define UA_NS0ID_CLOSESECURECHANNELRESPONSE_ENCODING_DEFAULTXML 454 // Object
#define UA_NS0ID_CLOSESECURECHANNELRESPONSE_ENCODING_DEFAULTBINARY 455 // Object
#define UA_NS0ID_SIGNATUREDATA 456 // DataType
#define UA_NS0ID_SIGNATUREDATA_ENCODING_DEFAULTXML 457 // Object
#define UA_NS0ID_SIGNATUREDATA_ENCODING_DEFAULTBINARY 458 // Object
#define UA_NS0ID_CREATESESSIONREQUEST 459 // DataType
#define UA_NS0ID_CREATESESSIONREQUEST_ENCODING_DEFAULTXML 460 // Object
#define UA_NS0ID_CREATESESSIONREQUEST_ENCODING_DEFAULTBINARY 461 // Object
#define UA_NS0ID_CREATESESSIONRESPONSE 462 // DataType
#define UA_NS0ID_CREATESESSIONRESPONSE_ENCODING_DEFAULTXML 463 // Object
#define UA_NS0ID_CREATESESSIONRESPONSE_ENCODING_DEFAULTBINARY 464 // Object
```

```
#define UA_NS0ID_ACTIVATESESSIONREQUEST 465 // DataType
#define UA_NS0ID_ACTIVATESESSIONREQUEST_ENCODING_DEFAULTXML 466 // Object
#define UA_NS0ID_ACTIVATESESSIONREQUEST_ENCODING_DEFAULTBINARY 467 // Object
#define UA_NS0ID_ACTIVATESESSIONRESPONSE 468 // DataType
#define UA_NS0ID_ACTIVATESESSIONRESPONSE_ENCODING_DEFAULTXML 469 // Object
#define UA_NS0ID_ACTIVATESESSIONRESPONSE_ENCODING_DEFAULTBINARY 470 // Object
#define UA_NS0ID_CLOSESESSIONREQUEST 471 // DataType
#define UA_NS0ID_CLOSESESSIONREQUEST_ENCODING_DEFAULTXML 472 // Object
#define UA_NS0ID_CLOSESESSIONREQUEST_ENCODING_DEFAULTBINARY 473 // Object
#define UA_NS0ID_CLOSESESSIONRESPONSE 474 // DataType
#define UA_NS0ID_CLOSESESSIONRESPONSE_ENCODING_DEFAULTXML 475 // Object
#define UA_NS0ID_CLOSESESSIONRESPONSE_ENCODING_DEFAULTBINARY 476 // Object
#define UA_NS0ID_CANCELREQUEST 477 // DataType
#define UA_NS0ID_CANCELREQUEST_ENCODING_DEFAULTXML 478 // Object
#define UA_NS0ID_CANCELREQUEST_ENCODING_DEFAULTBINARY 479 // Object
#define UA_NS0ID_CANCELRESPONSE 480 // DataType
#define UA_NS0ID_CANCELRESPONSE_ENCODING_DEFAULTXML 481 // Object
#define UA_NS0ID_CANCELRESPONSE_ENCODING_DEFAULTBINARY 482 // Object
#define UA_NS0ID_ADDNODESRESULT 483 // DataType
#define UA_NS0ID_ADDNODESRESULT_ENCODING_DEFAULTXML 484 // Object
#define UA_NS0ID_ADDNODESRESULT_ENCODING_DEFAULTBINARY 485 // Object
#define UA_NS0ID_ADDNODESREQUEST 486 // DataType
#define UA_NS0ID_ADDNODESREQUEST_ENCODING_DEFAULTXML 487 // Object
#define UA_NS0ID_ADDNODESREQUEST_ENCODING_DEFAULTBINARY 488 // Object
#define UA_NS0ID_ADDNODESRESPONSE 489 // DataType
#define UA_NS0ID_ADDNODESRESPONSE_ENCODING_DEFAULTXML 490 // Object
#define UA_NS0ID_ADDNODESRESPONSE_ENCODING_DEFAULTBINARY 491 // Object
#define UA_NS0ID_ADDREFERENCESREQUEST 492 // DataType
#define UA_NS0ID_ADDREFERENCESREQUEST_ENCODING_DEFAULTXML 493 // Object
#define UA_NS0ID_ADDREFERENCESREQUEST_ENCODING_DEFAULTBINARY 494 // Object
#define UA_NS0ID_ADDREFERENCESRESPONSE 495 // DataType
#define UA_NS0ID_ADDREFERENCESRESPONSE_ENCODING_DEFAULTXML 496 // Object
#define UA_NS0ID_ADDREFERENCESRESPONSE_ENCODING_DEFAULTBINARY 497 // Object
#define UA_NS0ID_DELETENODESREQUEST 498 // DataType
#define UA_NS0ID_DELETENODESREQUEST_ENCODING_DEFAULTXML 499 // Object
#define UA_NS0ID_DELETENODESREQUEST_ENCODING_DEFAULTBINARY 500 // Object
#define UA_NS0ID_DELETENODESRESPONSE 501 // DataType
#define UA_NS0ID_DELETENODESRESPONSE_ENCODING_DEFAULTXML 502 // Object
#define UA_NS0ID_DELETENODESRESPONSE_ENCODING_DEFAULTBINARY 503 // Object
#define UA_NS0ID_DELETEREFERENCESREQUEST 504 // DataType
#define UA_NS0ID_DELETEREFERENCESREQUEST_ENCODING_DEFAULTXML 505 // Object
#define UA_NS0ID_DELETEREFERENCESREQUEST_ENCODING_DEFAULTBINARY 506 // Object
#define UA_NS0ID_DELETEREFERENCESRESPONSE 507 // DataType
#define UA_NS0ID_DELETEREFERENCESRESPONSE_ENCODING_DEFAULTXML 508 // Object
#define UA_NS0ID_DELETEREFERENCESRESPONSE_ENCODING_DEFAULTBINARY 509 // Object
#define UA_NS0ID_BROWSEDIRECTION 510 // DataType
#define UA_NS0ID_VIEWDESCRIPTION 511 // DataType
#define UA_NS0ID_VIEWDESCRIPTION_ENCODING_DEFAULTXML 512 // Object
#define UA_NS0ID_VIEWDESCRIPTION_ENCODING_DEFAULTBINARY 513 // Object
#define UA_NS0ID_BROWSEDESCRIPTION 514 // DataType
#define UA_NS0ID_BROWSEDESCRIPTION_ENCODING_DEFAULTXML 515 // Object
#define UA_NS0ID_BROWSEDESCRIPTION_ENCODING_DEFAULTBINARY 516 // Object
#define UA_NS0ID_BROWSERESULTMASK 517 // DataType
#define UA_NS0ID_REFERENCEDESCRIPTION 518 // DataType
#define UA_NS0ID_REFERENCEDESCRIPTION_ENCODING_DEFAULTXML 519 // Object
#define UA_NS0ID_REFERENCEDESCRIPTION_ENCODING_DEFAULTBINARY 520 // Object
#define UA_NS0ID_CONTINUATIONPOINT 521 // DataType
#define UA_NS0ID_BROWSERESULT 522 // DataType
#define UA_NS0ID_BROWSERESULT_ENCODING_DEFAULTXML 523 // Object
#define UA_NS0ID_BROWSERESULT_ENCODING_DEFAULTBINARY 524 // Object
#define UA_NS0ID_BROWSEREQUEST 525 // DataType
```

```
#define UA_NS0ID_BROWSEREQUEST_ENCODING_DEFAULTXML 526 // Object
#define UA_NS0ID_BROWSEREQUEST_ENCODING_DEFAULTBINARY 527 // Object
#define UA_NS0ID_BROWSERESPONSE 528 // DataType
#define UA_NS0ID_BROWSERESPONSE_ENCODING_DEFAULTXML 529 // Object
#define UA_NS0ID_BROWSERESPONSE_ENCODING_DEFAULTBINARY 530 // Object
#define UA_NS0ID_BROWSENEXTREQUEST 531 // DataType
#define UA_NS0ID_BROWSENEXTREQUEST_ENCODING_DEFAULTXML 532 // Object
#define UA_NS0ID_BROWSENEXTREQUEST_ENCODING_DEFAULTBINARY 533 // Object
#define UA_NS0ID_BROWSENEXTRESPONSE 534 // DataType
#define UA_NS0ID_BROWSENEXTRESPONSE_ENCODING_DEFAULTXML 535 // Object
#define UA_NS0ID_BROWSENEXTRESPONSE_ENCODING_DEFAULTBINARY 536 // Object
#define UA_NS0ID_RELATIVEPATHELEMENT 537 // DataType
#define UA_NS0ID_RELATIVEPATHELEMENT_ENCODING_DEFAULTXML 538 // Object
#define UA_NS0ID_RELATIVEPATHELEMENT_ENCODING_DEFAULTBINARY 539 // Object
#define UA_NS0ID_RELATIVEPATH 540 // DataType
#define UA_NS0ID_RELATIVEPATH_ENCODING_DEFAULTXML 541 // Object
#define UA_NS0ID_RELATIVEPATH_ENCODING_DEFAULTBINARY 542 // Object
#define UA_NS0ID_BROWSEPATH 543 // DataType
#define UA_NS0ID_BROWSEPATH_ENCODING_DEFAULTXML 544 // Object
#define UA_NS0ID_BROWSEPATH_ENCODING_DEFAULTBINARY 545 // Object
#define UA_NS0ID_BROWSEPATHTARGET 546 // DataType
#define UA_NS0ID_BROWSEPATHTARGET_ENCODING_DEFAULTXML 547 // Object
#define UA_NS0ID_BROWSEPATHTARGET_ENCODING_DEFAULTBINARY 548 // Object
#define UA_NS0ID_BROWSEPATHRESULT 549 // DataType
#define UA_NS0ID_BROWSEPATHRESULT_ENCODING_DEFAULTXML 550 // Object
#define UA_NS0ID_BROWSEPATHRESULT_ENCODING_DEFAULTBINARY 551 // Object
#define UA_NS0ID_TRANSLATEBROWSEPATHSTONODEIDSREQUEST 552 // DataType
#define UA_NS0ID_TRANSLATEBROWSEPATHSTONODEIDSREQUEST_ENCODING_DEFAULTXML 553 //␣
→Object
#define UA_NS0ID_TRANSLATEBROWSEPATHSTONODEIDSREQUEST_ENCODING_DEFAULTBINARY 554 //
→ Object
#define UA_NS0ID_TRANSLATEBROWSEPATHSTONODEIDSRESPONSE 555 // DataType
#define UA_NS0ID_TRANSLATEBROWSEPATHSTONODEIDSRESPONSE_ENCODING_DEFAULTXML 556 //␣
→Object
#define UA_NS0ID_TRANSLATEBROWSEPATHSTONODEIDSRESPONSE_ENCODING_DEFAULTBINARY 557 /
→/ Object
#define UA_NS0ID_REGISTERNODESREQUEST 558 // DataType
#define UA_NS0ID_REGISTERNODESREQUEST_ENCODING_DEFAULTXML 559 // Object
#define UA_NS0ID_REGISTERNODESREQUEST_ENCODING_DEFAULTBINARY 560 // Object
#define UA_NS0ID_REGISTERNODESRESPONSE 561 // DataType
#define UA_NS0ID_REGISTERNODESRESPONSE_ENCODING_DEFAULTXML 562 // Object
#define UA_NS0ID_REGISTERNODESRESPONSE_ENCODING_DEFAULTBINARY 563 // Object
#define UA_NS0ID_UNREGISTERNODESREQUEST 564 // DataType
#define UA_NS0ID_UNREGISTERNODESREQUEST_ENCODING_DEFAULTXML 565 // Object
#define UA_NS0ID_UNREGISTERNODESREQUEST_ENCODING_DEFAULTBINARY 566 // Object
#define UA_NS0ID_UNREGISTERNODESRESPONSE 567 // DataType
#define UA_NS0ID_UNREGISTERNODESRESPONSE_ENCODING_DEFAULTXML 568 // Object
#define UA_NS0ID_UNREGISTERNODESRESPONSE_ENCODING_DEFAULTBINARY 569 // Object
#define UA_NS0ID_QUERYDATADESCRIPTION 570 // DataType
#define UA_NS0ID_QUERYDATADESCRIPTION_ENCODING_DEFAULTXML 571 // Object
#define UA_NS0ID_QUERYDATADESCRIPTION_ENCODING_DEFAULTBINARY 572 // Object
#define UA_NS0ID_NODETYPEDESCRIPTION 573 // DataType
#define UA_NS0ID_NODETYPEDESCRIPTION_ENCODING_DEFAULTXML 574 // Object
#define UA_NS0ID_NODETYPEDESCRIPTION_ENCODING_DEFAULTBINARY 575 // Object
#define UA_NS0ID_FILTEROPERATOR 576 // DataType
#define UA_NS0ID_QUERYDATASET 577 // DataType
#define UA_NS0ID_QUERYDATASET_ENCODING_DEFAULTXML 578 // Object
#define UA_NS0ID_QUERYDATASET_ENCODING_DEFAULTBINARY 579 // Object
#define UA_NS0ID_NODEREFERENCE 580 // DataType
#define UA_NS0ID_NODEREFERENCE_ENCODING_DEFAULTXML 581 // Object
#define UA_NS0ID_NODEREFERENCE_ENCODING_DEFAULTBINARY 582 // Object
```

```
#define UA_NS0ID_CONTENTFILTERELEMENT 583 // DataType
#define UA_NS0ID_CONTENTFILTERELEMENT_ENCODING_DEFAULTXML 584 // Object
#define UA_NS0ID_CONTENTFILTERELEMENT_ENCODING_DEFAULTBINARY 585 // Object
#define UA_NS0ID_CONTENTFILTER 586 // DataType
#define UA_NS0ID_CONTENTFILTER_ENCODING_DEFAULTXML 587 // Object
#define UA_NS0ID_CONTENTFILTER_ENCODING_DEFAULTBINARY 588 // Object
#define UA_NS0ID_FILTEROPERAND 589 // DataType
#define UA_NS0ID_FILTEROPERAND_ENCODING_DEFAULTXML 590 // Object
#define UA_NS0ID_FILTEROPERAND_ENCODING_DEFAULTBINARY 591 // Object
#define UA_NS0ID_ELEMENTOPERAND 592 // DataType
#define UA_NS0ID_ELEMENTOPERAND_ENCODING_DEFAULTXML 593 // Object
#define UA_NS0ID_ELEMENTOPERAND_ENCODING_DEFAULTBINARY 594 // Object
#define UA_NS0ID_LITERALOPERAND 595 // DataType
#define UA_NS0ID_LITERALOPERAND_ENCODING_DEFAULTXML 596 // Object
#define UA_NS0ID_LITERALOPERAND_ENCODING_DEFAULTBINARY 597 // Object
#define UA_NS0ID_ATTRIBUTEOPERAND 598 // DataType
#define UA_NS0ID_ATTRIBUTEOPERAND_ENCODING_DEFAULTXML 599 // Object
#define UA_NS0ID_ATTRIBUTEOPERAND_ENCODING_DEFAULTBINARY 600 // Object
#define UA_NS0ID_SIMPLEATTRIBUTEOPERAND 601 // DataType
#define UA_NS0ID_SIMPLEATTRIBUTEOPERAND_ENCODING_DEFAULTXML 602 // Object
#define UA_NS0ID_SIMPLEATTRIBUTEOPERAND_ENCODING_DEFAULTBINARY 603 // Object
#define UA_NS0ID_CONTENTFILTERELEMENTRESULT 604 // DataType
#define UA_NS0ID_CONTENTFILTERELEMENTRESULT_ENCODING_DEFAULTXML 605 // Object
#define UA_NS0ID_CONTENTFILTERELEMENTRESULT_ENCODING_DEFAULTBINARY 606 // Object
#define UA_NS0ID_CONTENTFILTERRESULT 607 // DataType
#define UA_NS0ID_CONTENTFILTERRESULT_ENCODING_DEFAULTXML 608 // Object
#define UA_NS0ID_CONTENTFILTERRESULT_ENCODING_DEFAULTBINARY 609 // Object
#define UA_NS0ID_PARSINGRESULT 610 // DataType
#define UA_NS0ID_PARSINGRESULT_ENCODING_DEFAULTXML 611 // Object
#define UA_NS0ID_PARSINGRESULT_ENCODING_DEFAULTBINARY 612 // Object
#define UA_NS0ID_QUERYFIRSTREQUEST 613 // DataType
#define UA_NS0ID_QUERYFIRSTREQUEST_ENCODING_DEFAULTXML 614 // Object
#define UA_NS0ID_QUERYFIRSTREQUEST_ENCODING_DEFAULTBINARY 615 // Object
#define UA_NS0ID_QUERYFIRSTRESPONSE 616 // DataType
#define UA_NS0ID_QUERYFIRSTRESPONSE_ENCODING_DEFAULTXML 617 // Object
#define UA_NS0ID_QUERYFIRSTRESPONSE_ENCODING_DEFAULTBINARY 618 // Object
#define UA_NS0ID_QUERYNEXTREQUEST 619 // DataType
#define UA_NS0ID_QUERYNEXTREQUEST_ENCODING_DEFAULTXML 620 // Object
#define UA_NS0ID_QUERYNEXTREQUEST_ENCODING_DEFAULTBINARY 621 // Object
#define UA_NS0ID_QUERYNEXTRESPONSE 622 // DataType
#define UA_NS0ID_QUERYNEXTRESPONSE_ENCODING_DEFAULTXML 623 // Object
#define UA_NS0ID_QUERYNEXTRESPONSE_ENCODING_DEFAULTBINARY 624 // Object
#define UA_NS0ID_TIMESTAMPSTORETURN 625 // DataType
#define UA_NS0ID_READVALUEID 626 // DataType
#define UA_NS0ID_READVALUEID_ENCODING_DEFAULTXML 627 // Object
#define UA_NS0ID_READVALUEID_ENCODING_DEFAULTBINARY 628 // Object
#define UA_NS0ID_READREQUEST 629 // DataType
#define UA_NS0ID_READREQUEST_ENCODING_DEFAULTXML 630 // Object
#define UA_NS0ID_READREQUEST_ENCODING_DEFAULTBINARY 631 // Object
#define UA_NS0ID_READRESPONSE 632 // DataType
#define UA_NS0ID_READRESPONSE_ENCODING_DEFAULTXML 633 // Object
#define UA_NS0ID_READRESPONSE_ENCODING_DEFAULTBINARY 634 // Object
#define UA_NS0ID_HISTORYREADVALUEID 635 // DataType
#define UA_NS0ID_HISTORYREADVALUEID_ENCODING_DEFAULTXML 636 // Object
#define UA_NS0ID_HISTORYREADVALUEID_ENCODING_DEFAULTBINARY 637 // Object
#define UA_NS0ID_HISTORYREADRESULT 638 // DataType
#define UA_NS0ID_HISTORYREADRESULT_ENCODING_DEFAULTXML 639 // Object
#define UA_NS0ID_HISTORYREADRESULT_ENCODING_DEFAULTBINARY 640 // Object
#define UA_NS0ID_HISTORYREADDETAILS 641 // DataType
#define UA_NS0ID_HISTORYREADDETAILS_ENCODING_DEFAULTXML 642 // Object
#define UA_NS0ID_HISTORYREADDETAILS_ENCODING_DEFAULTBINARY 643 // Object
```

```
#define UA_NS0ID_READEVENTDETAILS 644 // DataType
#define UA_NS0ID_READEVENTDETAILS_ENCODING_DEFAULTXML 645 // Object
#define UA_NS0ID_READEVENTDETAILS_ENCODING_DEFAULTBINARY 646 // Object
#define UA_NS0ID_READRAWMODIFIEDDETAILS 647 // DataType
#define UA_NS0ID_READRAWMODIFIEDDETAILS_ENCODING_DEFAULTXML 648 // Object
#define UA_NS0ID_READRAWMODIFIEDDETAILS_ENCODING_DEFAULTBINARY 649 // Object
#define UA_NS0ID_READPROCESSEDDETAILS 650 // DataType
#define UA_NS0ID_READPROCESSEDDETAILS_ENCODING_DEFAULTXML 651 // Object
#define UA_NS0ID_READPROCESSEDDETAILS_ENCODING_DEFAULTBINARY 652 // Object
#define UA_NS0ID_READATTIMEDETAILS 653 // DataType
#define UA_NS0ID_READATTIMEDETAILS_ENCODING_DEFAULTXML 654 // Object
#define UA_NS0ID_READATTIMEDETAILS_ENCODING_DEFAULTBINARY 655 // Object
#define UA_NS0ID_HISTORYDATA 656 // DataType
#define UA_NS0ID_HISTORYDATA_ENCODING_DEFAULTXML 657 // Object
#define UA_NS0ID_HISTORYDATA_ENCODING_DEFAULTBINARY 658 // Object
#define UA_NS0ID_HISTORYEVENT 659 // DataType
#define UA_NS0ID_HISTORYEVENT_ENCODING_DEFAULTXML 660 // Object
#define UA_NS0ID_HISTORYEVENT_ENCODING_DEFAULTBINARY 661 // Object
#define UA_NS0ID_HISTORYREADREQUEST 662 // DataType
#define UA_NS0ID_HISTORYREADREQUEST_ENCODING_DEFAULTXML 663 // Object
#define UA_NS0ID_HISTORYREADREQUEST_ENCODING_DEFAULTBINARY 664 // Object
#define UA_NS0ID_HISTORYREADRESPONSE 665 // DataType
#define UA_NS0ID_HISTORYREADRESPONSE_ENCODING_DEFAULTXML 666 // Object
#define UA_NS0ID_HISTORYREADRESPONSE_ENCODING_DEFAULTBINARY 667 // Object
#define UA_NS0ID_WRITEVALUE 668 // DataType
#define UA_NS0ID_WRITEVALUE_ENCODING_DEFAULTXML 669 // Object
#define UA_NS0ID_WRITEVALUE_ENCODING_DEFAULTBINARY 670 // Object
#define UA_NS0ID_WRITEREQUEST 671 // DataType
#define UA_NS0ID_WRITEREQUEST_ENCODING_DEFAULTXML 672 // Object
#define UA_NS0ID_WRITEREQUEST_ENCODING_DEFAULTBINARY 673 // Object
#define UA_NS0ID_WRITERESPONSE 674 // DataType
#define UA_NS0ID_WRITERESPONSE_ENCODING_DEFAULTXML 675 // Object
#define UA_NS0ID_WRITERESPONSE_ENCODING_DEFAULTBINARY 676 // Object
#define UA_NS0ID_HISTORYUPDATEDETAILS 677 // DataType
#define UA_NS0ID_HISTORYUPDATEDETAILS_ENCODING_DEFAULTXML 678 // Object
#define UA_NS0ID_HISTORYUPDATEDETAILS_ENCODING_DEFAULTBINARY 679 // Object
#define UA_NS0ID_UPDATEDATADETAILS 680 // DataType
#define UA_NS0ID_UPDATEDATADETAILS_ENCODING_DEFAULTXML 681 // Object
#define UA_NS0ID_UPDATEDATADETAILS_ENCODING_DEFAULTBINARY 682 // Object
#define UA_NS0ID_UPDATEEVENTDETAILS 683 // DataType
#define UA_NS0ID_UPDATEEVENTDETAILS_ENCODING_DEFAULTXML 684 // Object
#define UA_NS0ID_UPDATEEVENTDETAILS_ENCODING_DEFAULTBINARY 685 // Object
#define UA_NS0ID_DELETERAWMODIFIEDDETAILS 686 // DataType
#define UA_NS0ID_DELETERAWMODIFIEDDETAILS_ENCODING_DEFAULTXML 687 // Object
#define UA_NS0ID_DELETERAWMODIFIEDDETAILS_ENCODING_DEFAULTBINARY 688 // Object
#define UA_NS0ID_DELETEATTIMEDETAILS 689 // DataType
#define UA_NS0ID_DELETEATTIMEDETAILS_ENCODING_DEFAULTXML 690 // Object
#define UA_NS0ID_DELETEATTIMEDETAILS_ENCODING_DEFAULTBINARY 691 // Object
#define UA_NS0ID_DELETEEVENTDETAILS 692 // DataType
#define UA_NS0ID_DELETEEVENTDETAILS_ENCODING_DEFAULTXML 693 // Object
#define UA_NS0ID_DELETEEVENTDETAILS_ENCODING_DEFAULTBINARY 694 // Object
#define UA_NS0ID_HISTORYUPDATERESULT 695 // DataType
#define UA_NS0ID_HISTORYUPDATERESULT_ENCODING_DEFAULTXML 696 // Object
#define UA_NS0ID_HISTORYUPDATERESULT_ENCODING_DEFAULTBINARY 697 // Object
#define UA_NS0ID_HISTORYUPDATEREQUEST 698 // DataType
#define UA_NS0ID_HISTORYUPDATEREQUEST_ENCODING_DEFAULTXML 699 // Object
#define UA_NS0ID_HISTORYUPDATEREQUEST_ENCODING_DEFAULTBINARY 700 // Object
#define UA_NS0ID_HISTORYUPDATERESPONSE 701 // DataType
#define UA_NS0ID_HISTORYUPDATERESPONSE_ENCODING_DEFAULTXML 702 // Object
#define UA_NS0ID_HISTORYUPDATERESPONSE_ENCODING_DEFAULTBINARY 703 // Object
#define UA_NS0ID_CALLMETHODREQUEST 704 // DataType
```

```
#define UA_NS0ID_CALLMETHODREQUEST_ENCODING_DEFAULTXML 705 // Object
#define UA_NS0ID_CALLMETHODREQUEST_ENCODING_DEFAULTBINARY 706 // Object
#define UA_NS0ID_CALLMETHODRESULT 707 // DataType
#define UA_NS0ID_CALLMETHODRESULT_ENCODING_DEFAULTXML 708 // Object
#define UA_NS0ID_CALLMETHODRESULT_ENCODING_DEFAULTBINARY 709 // Object
#define UA_NS0ID_CALLREQUEST 710 // DataType
#define UA_NS0ID_CALLREQUEST_ENCODING_DEFAULTXML 711 // Object
#define UA_NS0ID_CALLREQUEST_ENCODING_DEFAULTBINARY 712 // Object
#define UA_NS0ID_CALLRESPONSE 713 // DataType
#define UA_NS0ID_CALLRESPONSE_ENCODING_DEFAULTXML 714 // Object
#define UA_NS0ID_CALLRESPONSE_ENCODING_DEFAULTBINARY 715 // Object
#define UA_NS0ID_MONITORINGMODE 716 // DataType
#define UA_NS0ID_DATACHANGETRIGGER 717 // DataType
#define UA_NS0ID_DEADBANDTYPE 718 // DataType
#define UA_NS0ID_MONITORINGFILTER 719 // DataType
#define UA_NS0ID_MONITORINGFILTER_ENCODING_DEFAULTXML 720 // Object
#define UA_NS0ID_MONITORINGFILTER_ENCODING_DEFAULTBINARY 721 // Object
#define UA_NS0ID_DATACHANGEFILTER 722 // DataType
#define UA_NS0ID_DATACHANGEFILTER_ENCODING_DEFAULTXML 723 // Object
#define UA_NS0ID_DATACHANGEFILTER_ENCODING_DEFAULTBINARY 724 // Object
#define UA_NS0ID_EVENTFILTER 725 // DataType
#define UA_NS0ID_EVENTFILTER_ENCODING_DEFAULTXML 726 // Object
#define UA_NS0ID_EVENTFILTER_ENCODING_DEFAULTBINARY 727 // Object
#define UA_NS0ID_AGGREGATEFILTER 728 // DataType
#define UA_NS0ID_AGGREGATEFILTER_ENCODING_DEFAULTXML 729 // Object
#define UA_NS0ID_AGGREGATEFILTER_ENCODING_DEFAULTBINARY 730 // Object
#define UA_NS0ID_MONITORINGFILTERRESULT 731 // DataType
#define UA_NS0ID_MONITORINGFILTERRESULT_ENCODING_DEFAULTXML 732 // Object
#define UA_NS0ID_MONITORINGFILTERRESULT_ENCODING_DEFAULTBINARY 733 // Object
#define UA_NS0ID_EVENTFILTERRESULT 734 // DataType
#define UA_NS0ID_EVENTFILTERRESULT_ENCODING_DEFAULTXML 735 // Object
#define UA_NS0ID_EVENTFILTERRESULT_ENCODING_DEFAULTBINARY 736 // Object
#define UA_NS0ID_AGGREGATEFILTERRESULT 737 // DataType
#define UA_NS0ID_AGGREGATEFILTERRESULT_ENCODING_DEFAULTXML 738 // Object
#define UA_NS0ID_AGGREGATEFILTERRESULT_ENCODING_DEFAULTBINARY 739 // Object
#define UA_NS0ID_MONITORINGPARAMETERS 740 // DataType
#define UA_NS0ID_MONITORINGPARAMETERS_ENCODING_DEFAULTXML 741 // Object
#define UA_NS0ID_MONITORINGPARAMETERS_ENCODING_DEFAULTBINARY 742 // Object
#define UA_NS0ID_MONITOREDITEMCREATEREQUEST 743 // DataType
#define UA_NS0ID_MONITOREDITEMCREATEREQUEST_ENCODING_DEFAULTXML 744 // Object
#define UA_NS0ID_MONITOREDITEMCREATEREQUEST_ENCODING_DEFAULTBINARY 745 // Object
#define UA_NS0ID_MONITOREDITEMCREATERESULT 746 // DataType
#define UA_NS0ID_MONITOREDITEMCREATERESULT_ENCODING_DEFAULTXML 747 // Object
#define UA_NS0ID_MONITOREDITEMCREATERESULT_ENCODING_DEFAULTBINARY 748 // Object
#define UA_NS0ID_CREATEMONITOREDITEMSREQUEST 749 // DataType
#define UA_NS0ID_CREATEMONITOREDITEMSREQUEST_ENCODING_DEFAULTXML 750 // Object
#define UA_NS0ID_CREATEMONITOREDITEMSREQUEST_ENCODING_DEFAULTBINARY 751 // Object
#define UA_NS0ID_CREATEMONITOREDITEMSRESPONSE 752 // DataType
#define UA_NS0ID_CREATEMONITOREDITEMSRESPONSE_ENCODING_DEFAULTXML 753 // Object
#define UA_NS0ID_CREATEMONITOREDITEMSRESPONSE_ENCODING_DEFAULTBINARY 754 // Object
#define UA_NS0ID_MONITOREDITEMMODIFYREQUEST 755 // DataType
#define UA_NS0ID_MONITOREDITEMMODIFYREQUEST_ENCODING_DEFAULTXML 756 // Object
#define UA_NS0ID_MONITOREDITEMMODIFYREQUEST_ENCODING_DEFAULTBINARY 757 // Object
#define UA_NS0ID_MONITOREDITEMMODIFYRESULT 758 // DataType
#define UA_NS0ID_MONITOREDITEMMODIFYRESULT_ENCODING_DEFAULTXML 759 // Object
#define UA_NS0ID_MONITOREDITEMMODIFYRESULT_ENCODING_DEFAULTBINARY 760 // Object
#define UA_NS0ID_MODIFYMONITOREDITEMSREQUEST 761 // DataType
#define UA_NS0ID_MODIFYMONITOREDITEMSREQUEST_ENCODING_DEFAULTXML 762 // Object
#define UA_NS0ID_MODIFYMONITOREDITEMSREQUEST_ENCODING_DEFAULTBINARY 763 // Object
#define UA_NS0ID_MODIFYMONITOREDITEMSRESPONSE 764 // DataType
#define UA_NS0ID_MODIFYMONITOREDITEMSRESPONSE_ENCODING_DEFAULTXML 765 // Object
```

```
#define UA_NS0ID_MODIFYMONITOREDITEMSRESPONSE_ENCODING_DEFAULTBINARY 766 // Object
#define UA_NS0ID_SETMONITORINGMODEREQUEST 767 // DataType
#define UA_NS0ID_SETMONITORINGMODEREQUEST_ENCODING_DEFAULTXML 768 // Object
#define UA_NS0ID_SETMONITORINGMODEREQUEST_ENCODING_DEFAULTBINARY 769 // Object
#define UA_NS0ID_SETMONITORINGMODERESPONSE 770 // DataType
#define UA_NS0ID_SETMONITORINGMODERESPONSE_ENCODING_DEFAULTXML 771 // Object
#define UA_NS0ID_SETMONITORINGMODERESPONSE_ENCODING_DEFAULTBINARY 772 // Object
#define UA_NS0ID_SETTRIGGERINGREQUEST 773 // DataType
#define UA_NS0ID_SETTRIGGERINGREQUEST_ENCODING_DEFAULTXML 774 // Object
#define UA_NS0ID_SETTRIGGERINGREQUEST_ENCODING_DEFAULTBINARY 775 // Object
#define UA_NS0ID_SETTRIGGERINGRESPONSE 776 // DataType
#define UA_NS0ID_SETTRIGGERINGRESPONSE_ENCODING_DEFAULTXML 777 // Object
#define UA_NS0ID_SETTRIGGERINGRESPONSE_ENCODING_DEFAULTBINARY 778 // Object
#define UA_NS0ID_DELETEMONITOREDITEMSREQUEST 779 // DataType
#define UA_NS0ID_DELETEMONITOREDITEMSREQUEST_ENCODING_DEFAULTXML 780 // Object
#define UA_NS0ID_DELETEMONITOREDITEMSREQUEST_ENCODING_DEFAULTBINARY 781 // Object
#define UA_NS0ID_DELETEMONITOREDITEMSRESPONSE 782 // DataType
#define UA_NS0ID_DELETEMONITOREDITEMSRESPONSE_ENCODING_DEFAULTXML 783 // Object
#define UA_NS0ID_DELETEMONITOREDITEMSRESPONSE_ENCODING_DEFAULTBINARY 784 // Object
#define UA_NS0ID_CREATESUBSCRIPTIONREQUEST 785 // DataType
#define UA_NS0ID_CREATESUBSCRIPTIONREQUEST_ENCODING_DEFAULTXML 786 // Object
#define UA_NS0ID_CREATESUBSCRIPTIONREQUEST_ENCODING_DEFAULTBINARY 787 // Object
#define UA_NS0ID_CREATESUBSCRIPTIONRESPONSE 788 // DataType
#define UA_NS0ID_CREATESUBSCRIPTIONRESPONSE_ENCODING_DEFAULTXML 789 // Object
#define UA_NS0ID_CREATESUBSCRIPTIONRESPONSE_ENCODING_DEFAULTBINARY 790 // Object
#define UA_NS0ID_MODIFYSUBSCRIPTIONREQUEST 791 // DataType
#define UA_NS0ID_MODIFYSUBSCRIPTIONREQUEST_ENCODING_DEFAULTXML 792 // Object
#define UA_NS0ID_MODIFYSUBSCRIPTIONREQUEST_ENCODING_DEFAULTBINARY 793 // Object
#define UA_NS0ID_MODIFYSUBSCRIPTIONRESPONSE 794 // DataType
#define UA_NS0ID_MODIFYSUBSCRIPTIONRESPONSE_ENCODING_DEFAULTXML 795 // Object
#define UA_NS0ID_MODIFYSUBSCRIPTIONRESPONSE_ENCODING_DEFAULTBINARY 796 // Object
#define UA_NS0ID_SETPUBLISHINGMODEREQUEST 797 // DataType
#define UA_NS0ID_SETPUBLISHINGMODEREQUEST_ENCODING_DEFAULTXML 798 // Object
#define UA_NS0ID_SETPUBLISHINGMODEREQUEST_ENCODING_DEFAULTBINARY 799 // Object
#define UA_NS0ID_SETPUBLISHINGMODERESPONSE 800 // DataType
#define UA_NS0ID_SETPUBLISHINGMODERESPONSE_ENCODING_DEFAULTXML 801 // Object
#define UA_NS0ID_SETPUBLISHINGMODERESPONSE_ENCODING_DEFAULTBINARY 802 // Object
#define UA_NS0ID_NOTIFICATIONMESSAGE 803 // DataType
#define UA_NS0ID_NOTIFICATIONMESSAGE_ENCODING_DEFAULTXML 804 // Object
#define UA_NS0ID_NOTIFICATIONMESSAGE_ENCODING_DEFAULTBINARY 805 // Object
#define UA_NS0ID_MONITOREDITEMNOTIFICATION 806 // DataType
#define UA_NS0ID_MONITOREDITEMNOTIFICATION_ENCODING_DEFAULTXML 807 // Object
#define UA_NS0ID_MONITOREDITEMNOTIFICATION_ENCODING_DEFAULTBINARY 808 // Object
#define UA_NS0ID_DATACHANGENOTIFICATION 809 // DataType
#define UA_NS0ID_DATACHANGENOTIFICATION_ENCODING_DEFAULTXML 810 // Object
#define UA_NS0ID_DATACHANGENOTIFICATION_ENCODING_DEFAULTBINARY 811 // Object
#define UA_NS0ID_STATUSCHANGENOTIFICATION 818 // DataType
#define UA_NS0ID_STATUSCHANGENOTIFICATION_ENCODING_DEFAULTXML 819 // Object
#define UA_NS0ID_STATUSCHANGENOTIFICATION_ENCODING_DEFAULTBINARY 820 // Object
#define UA_NS0ID_SUBSCRIPTIONACKNOWLEDGEMENT 821 // DataType
#define UA_NS0ID_SUBSCRIPTIONACKNOWLEDGEMENT_ENCODING_DEFAULTXML 822 // Object
#define UA_NS0ID_SUBSCRIPTIONACKNOWLEDGEMENT_ENCODING_DEFAULTBINARY 823 // Object
#define UA_NS0ID_PUBLISHREQUEST 824 // DataType
#define UA_NS0ID_PUBLISHREQUEST_ENCODING_DEFAULTXML 825 // Object
#define UA_NS0ID_PUBLISHREQUEST_ENCODING_DEFAULTBINARY 826 // Object
#define UA_NS0ID_PUBLISHRESPONSE 827 // DataType
#define UA_NS0ID_PUBLISHRESPONSE_ENCODING_DEFAULTXML 828 // Object
#define UA_NS0ID_PUBLISHRESPONSE_ENCODING_DEFAULTBINARY 829 // Object
#define UA_NS0ID_REPUBLISHREQUEST 830 // DataType
#define UA_NS0ID_REPUBLISHREQUEST_ENCODING_DEFAULTXML 831 // Object
#define UA_NS0ID_REPUBLISHREQUEST_ENCODING_DEFAULTBINARY 832 // Object
```

```
#define UA_NS0ID_REPUBLISHRESPONSE 833 // DataType
#define UA_NS0ID_REPUBLISHRESPONSE_ENCODING_DEFAULTXML 834 // Object
#define UA_NS0ID_REPUBLISHRESPONSE_ENCODING_DEFAULTBINARY 835 // Object
#define UA_NS0ID_TRANSFERRESULT 836 // DataType
#define UA_NS0ID_TRANSFERRESULT_ENCODING_DEFAULTXML 837 // Object
#define UA_NS0ID_TRANSFERRESULT_ENCODING_DEFAULTBINARY 838 // Object
#define UA_NS0ID_TRANSFERSUBSCRIPTIONSREQUEST 839 // DataType
#define UA_NS0ID_TRANSFERSUBSCRIPTIONSREQUEST_ENCODING_DEFAULTXML 840 // Object
#define UA_NS0ID_TRANSFERSUBSCRIPTIONSREQUEST_ENCODING_DEFAULTBINARY 841 // Object
#define UA_NS0ID_TRANSFERSUBSCRIPTIONSRESPONSE 842 // DataType
#define UA_NS0ID_TRANSFERSUBSCRIPTIONSRESPONSE_ENCODING_DEFAULTXML 843 // Object
#define UA_NS0ID_TRANSFERSUBSCRIPTIONSRESPONSE_ENCODING_DEFAULTBINARY 844 // Object
#define UA_NS0ID_DELETESUBSCRIPTIONSREQUEST 845 // DataType
#define UA_NS0ID_DELETESUBSCRIPTIONSREQUEST_ENCODING_DEFAULTXML 846 // Object
#define UA_NS0ID_DELETESUBSCRIPTIONSREQUEST_ENCODING_DEFAULTBINARY 847 // Object
#define UA_NS0ID_DELETESUBSCRIPTIONSRESPONSE 848 // DataType
#define UA_NS0ID_DELETESUBSCRIPTIONSRESPONSE_ENCODING_DEFAULTXML 849 // Object
#define UA_NS0ID_DELETESUBSCRIPTIONSRESPONSE_ENCODING_DEFAULTBINARY 850 // Object
#define UA_NS0ID_REDUNDANCYSUPPORT 851 // DataType
#define UA_NS0ID_SERVERSTATE 852 // DataType
#define UA_NS0ID_REDUNDANTSERVERDATATYPE 853 // DataType
#define UA_NS0ID_SAMPLINGINTERVALDIAGNOSTICSDATATYPE 856 // DataType
#define UA_NS0ID_SERVERDIAGNOSTICSSUMMARYDATATYPE 859 // DataType
#define UA_NS0ID_SERVERSTATUSDATATYPE 862 // DataType
#define UA_NS0ID_SESSIONDIAGNOSTICSDATATYPE 865 // DataType
#define UA_NS0ID_SESSIONSECURITYDIAGNOSTICSDATATYPE 868 // DataType
#define UA_NS0ID_SERVICECOUNTERDATATYPE 871 // DataType
#define UA_NS0ID_SUBSCRIPTIONDIAGNOSTICSDATATYPE 874 // DataType
#define UA_NS0ID_MODELCHANGESTRUCTUREDATATYPE 877 // DataType
#define UA_NS0ID_RANGE 884 // DataType
#define UA_NS0ID_RANGE_ENCODING_DEFAULTXML 885 // Object
#define UA_NS0ID_RANGE_ENCODING_DEFAULTBINARY 886 // Object
#define UA_NS0ID_EUINFORMATION 887 // DataType
#define UA_NS0ID_EUINFORMATION_ENCODING_DEFAULTXML 888 // Object
#define UA_NS0ID_EUINFORMATION_ENCODING_DEFAULTBINARY 889 // Object
#define UA_NS0ID_EXCEPTIONDEVIATIONFORMAT 890 // DataType
#define UA_NS0ID_ANNOTATION 891 // DataType
#define UA_NS0ID_ANNOTATION_ENCODING_DEFAULTXML 892 // Object
#define UA_NS0ID_ANNOTATION_ENCODING_DEFAULTBINARY 893 // Object
#define UA_NS0ID_PROGRAMDIAGNOSTICDATATYPE 894 // DataType
#define UA_NS0ID_SEMANTICCHANGESTRUCTUREDATATYPE 897 // DataType
#define UA_NS0ID_EVENTNOTIFICATIONLIST 914 // DataType
#define UA_NS0ID_EVENTNOTIFICATIONLIST_ENCODING_DEFAULTXML 915 // Object
#define UA_NS0ID_EVENTNOTIFICATIONLIST_ENCODING_DEFAULTBINARY 916 // Object
#define UA_NS0ID_EVENTFIELDLIST 917 // DataType
#define UA_NS0ID_EVENTFIELDLIST_ENCODING_DEFAULTXML 918 // Object
#define UA_NS0ID_EVENTFIELDLIST_ENCODING_DEFAULTBINARY 919 // Object
#define UA_NS0ID_HISTORYEVENTFIELDLIST 920 // DataType
#define UA_NS0ID_HISTORYEVENTFIELDLIST_ENCODING_DEFAULTXML 921 // Object
#define UA_NS0ID_HISTORYEVENTFIELDLIST_ENCODING_DEFAULTBINARY 922 // Object
#define UA_NS0ID_ISSUEDIDENTITYTOKEN 938 // DataType
#define UA_NS0ID_ISSUEDIDENTITYTOKEN_ENCODING_DEFAULTXML 939 // Object
#define UA_NS0ID_ISSUEDIDENTITYTOKEN_ENCODING_DEFAULTBINARY 940 // Object
#define UA_NS0ID_NOTIFICATIONDATA 945 // DataType
#define UA_NS0ID_NOTIFICATIONDATA_ENCODING_DEFAULTXML 946 // Object
#define UA_NS0ID_NOTIFICATIONDATA_ENCODING_DEFAULTBINARY 947 // Object
#define UA_NS0ID_AGGREGATECONFIGURATION 948 // DataType
#define UA_NS0ID_AGGREGATECONFIGURATION_ENCODING_DEFAULTXML 949 // Object
#define UA_NS0ID_AGGREGATECONFIGURATION_ENCODING_DEFAULTBINARY 950 // Object
#define UA_NS0ID_IMAGEBMP 2000 // DataType
#define UA_NS0ID_IMAGEGIF 2001 // DataType
```

```
#define UA_NS0ID_IMAGEJPG 2002 // DataType
#define UA_NS0ID_IMAGEPNG 2003 // DataType
#define UA_NS0ID_SERVERTYPE 2004 // ObjectType
#define UA_NS0ID_SERVERCAPABILITIESTYPE 2013 // ObjectType
#define UA_NS0ID_SERVERDIAGNOSTICSTYPE 2020 // ObjectType
#define UA_NS0ID_SESSIONSDIAGNOSTICSSUMMARYTYPE 2026 // ObjectType
#define UA_NS0ID_SESSIONDIAGNOSTICSOBJECTTYPE 2029 // ObjectType
#define UA_NS0ID_VENDORSERVERINFOTYPE 2033 // ObjectType
#define UA_NS0ID_SERVERREDUNDANCYTYPE 2034 // ObjectType
#define UA_NS0ID_TRANSPARENTREDUNDANCYTYPE 2036 // ObjectType
#define UA_NS0ID_NONTRANSPARENTREDUNDANCYTYPE 2039 // ObjectType
#define UA_NS0ID_BASEEVENTTYPE 2041 // ObjectType
#define UA_NS0ID_AUDITEVENTTYPE 2052 // ObjectType
#define UA_NS0ID_AUDITSECURITYEVENTTYPE 2058 // ObjectType
#define UA_NS0ID_AUDITCHANNELEVENTTYPE 2059 // ObjectType
#define UA_NS0ID_AUDITOPENSECURECHANNELEVENTTYPE 2060 // ObjectType
#define UA_NS0ID_AUDITSESSIONEVENTTYPE 2069 // ObjectType
#define UA_NS0ID_AUDITCREATESESSIONEVENTTYPE 2071 // ObjectType
#define UA_NS0ID_AUDITACTIVATESESSIONEVENTTYPE 2075 // ObjectType
#define UA_NS0ID_AUDITCANCELEVENTTYPE 2078 // ObjectType
#define UA_NS0ID_AUDITCERTIFICATEEVENTTYPE 2080 // ObjectType
#define UA_NS0ID_AUDITCERTIFICATEDATAMISMATCHEVENTTYPE 2082 // ObjectType
#define UA_NS0ID_AUDITCERTIFICATEEXPIREDEVENTTYPE 2085 // ObjectType
#define UA_NS0ID_AUDITCERTIFICATEINVALIDEVENTTYPE 2086 // ObjectType
#define UA_NS0ID_AUDITCERTIFICATEUNTRUSTEDEVENTTYPE 2087 // ObjectType
#define UA_NS0ID_AUDITCERTIFICATEREVOKEDEVENTTYPE 2088 // ObjectType
#define UA_NS0ID_AUDITCERTIFICATEMISMATCHEVENTTYPE 2089 // ObjectType
#define UA_NS0ID_AUDITNODEMANAGEMENTEVENTTYPE 2090 // ObjectType
#define UA_NS0ID_AUDITADDNODESEVENTTYPE 2091 // ObjectType
#define UA_NS0ID_AUDITDELETENODESEVENTTYPE 2093 // ObjectType
#define UA_NS0ID_AUDITADDREFERENCESEVENTTYPE 2095 // ObjectType
#define UA_NS0ID_AUDITDELETEREFERENCESEVENTTYPE 2097 // ObjectType
#define UA_NS0ID_AUDITUPDATEEVENTTYPE 2099 // ObjectType
#define UA_NS0ID_AUDITWRITEUPDATEEVENTTYPE 2100 // ObjectType
#define UA_NS0ID_AUDITHISTORYUPDATEEVENTTYPE 2104 // ObjectType
#define UA_NS0ID_AUDITUPDATEMETHODEVENTTYPE 2127 // ObjectType
#define UA_NS0ID_SYSTEMEVENTTYPE 2130 // ObjectType
#define UA_NS0ID_DEVICEFAILUREEVENTTYPE 2131 // ObjectType
#define UA_NS0ID_BASEMODELCHANGEEVENTTYPE 2132 // ObjectType
#define UA_NS0ID_GENERALMODELCHANGEEVENTTYPE 2133 // ObjectType
#define UA_NS0ID_SERVERVENDORCAPABILITYTYPE 2137 // VariableType
#define UA_NS0ID_SERVERSTATUSTYPE 2138 // VariableType
#define UA_NS0ID_SERVERDIAGNOSTICSSUMMARYTYPE 2150 // VariableType
#define UA_NS0ID_SAMPLINGINTERVALDIAGNOSTICSARRAYTYPE 2164 // VariableType
#define UA_NS0ID_SAMPLINGINTERVALDIAGNOSTICSTYPE 2165 // VariableType
#define UA_NS0ID_SUBSCRIPTIONDIAGNOSTICSARRAYTYPE 2171 // VariableType
#define UA_NS0ID_SUBSCRIPTIONDIAGNOSTICSTYPE 2172 // VariableType
#define UA_NS0ID_SESSIONDIAGNOSTICSARRAYTYPE 2196 // VariableType
#define UA_NS0ID_SESSIONDIAGNOSTICSVARIABLETYPE 2197 // VariableType
#define UA_NS0ID_SESSIONSECURITYDIAGNOSTICSARRAYTYPE 2243 // VariableType
#define UA_NS0ID_SESSIONSECURITYDIAGNOSTICSTYPE 2244 // VariableType
#define UA_NS0ID_SERVER 2253 // Object
#define UA_NS0ID_SERVER_SERVERARRAY 2254 // Variable
#define UA_NS0ID_SERVER_NAMESPACEARRAY 2255 // Variable
#define UA_NS0ID_SERVER_SERVERSTATUS 2256 // Variable
#define UA_NS0ID_SERVER_SERVERSTATUS_STARTTIME 2257 // Variable
#define UA_NS0ID_SERVER_SERVERSTATUS_CURRENTTIME 2258 // Variable
#define UA_NS0ID_SERVER_SERVERSTATUS_STATE 2259 // Variable
#define UA_NS0ID_SERVER_SERVERSTATUS_BUILDINFO 2260 // Variable
#define UA_NS0ID_SERVER_SERVERSTATUS_BUILDINFO_PRODUCTNAME 2261 // Variable
#define UA_NS0ID_SERVER_SERVERSTATUS_BUILDINFO_PRODUCTURI 2262 // Variable
```

```
#define UA_NS0ID_SERVER_SERVERSTATUS_BUILDINFO_MANUFACTURERNAME 2263 // Variable
#define UA_NS0ID_SERVER_SERVERSTATUS_BUILDINFO_SOFTWAREVERSION 2264 // Variable
#define UA_NS0ID_SERVER_SERVERSTATUS_BUILDINFO_BUILDNUMBER 2265 // Variable
#define UA_NS0ID_SERVER_SERVERSTATUS_BUILDINFO_BUILDDATE 2266 // Variable
#define UA_NS0ID_SERVER_SERVICELEVEL 2267 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES 2268 // Object
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_SERVERPROFILEARRAY 2269 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_LOCALEIDARRAY 2271 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_MINSUPPORTEDSAMPLERATE 2272 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS 2274 // Object
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY 2275 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_SERVERVIEWCOUNT␣
↪2276 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_
↪CURRENTSESSIONCOUNT 2277 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_
↪CUMULATEDSESSIONCOUNT 2278 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_
↪SECURITYREJECTEDSESSIONCOUNT 2279 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_
↪SESSIONTIMEOUTCOUNT 2281 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_
↪SESSIONABORTCOUNT 2282 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_
↪PUBLISHINGINTERVALCOUNT 2284 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_
↪CURRENTSUBSCRIPTIONCOUNT 2285 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_
↪CUMULATEDSUBSCRIPTIONCOUNT 2286 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_
↪SECURITYREJECTEDREQUESTSCOUNT 2287 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_
↪REJECTEDREQUESTSCOUNT 2288 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SAMPLINGINTERVALDIAGNOSTICSARRAY 2289 //␣
↪Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SUBSCRIPTIONDIAGNOSTICSARRAY 2290 //␣
↪Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_ENABLEDFLAG 2294 // Variable
#define UA_NS0ID_SERVER_VENDORSERVERINFO 2295 // Object
#define UA_NS0ID_SERVER_SERVERREDUNDANCY 2296 // Object
#define UA_NS0ID_STATEMACHINETYPE 2299 // ObjectType
#define UA_NS0ID_STATETYPE 2307 // ObjectType
#define UA_NS0ID_INITIALSTATETYPE 2309 // ObjectType
#define UA_NS0ID_TRANSITIONTYPE 2310 // ObjectType
#define UA_NS0ID_TRANSITIONEVENTTYPE 2311 // ObjectType
#define UA_NS0ID_AUDITUPDATESTATEEVENTTYPE 2315 // ObjectType
#define UA_NS0ID_HISTORICALDATACONFIGURATIONTYPE 2318 // ObjectType
#define UA_NS0ID_HISTORYSERVERCAPABILITIESTYPE 2330 // ObjectType
#define UA_NS0ID_AGGREGATEFUNCTIONTYPE 2340 // ObjectType
#define UA_NS0ID_AGGREGATEFUNCTION_INTERPOLATIVE 2341 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_AVERAGE 2342 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_TIMEAVERAGE 2343 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_TOTAL 2344 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_MINIMUM 2346 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_MAXIMUM 2347 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_MINIMUMACTUALTIME 2348 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_MAXIMUMACTUALTIME 2349 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_RANGE 2350 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_ANNOTATIONCOUNT 2351 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_COUNT 2352 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_NUMBEROFTRANSITIONS 2355 // Object
```

```
#define UA_NS0ID_AGGREGATEFUNCTION_START 2357 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_END 2358 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_DELTA 2359 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_DURATIONGOOD 2360 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_DURATIONBAD 2361 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_PERCENTGOOD 2362 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_PERCENTBAD 2363 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_WORSTQUALITY 2364 // Object
#define UA_NS0ID_DATAITEMTYPE 2365 // VariableType
#define UA_NS0ID_ANALOGITEMTYPE 2368 // VariableType
#define UA_NS0ID_DISCRETEITEMTYPE 2372 // VariableType
#define UA_NS0ID_TWOSTATEDISCRETETYPE 2373 // VariableType
#define UA_NS0ID_MULTISTATEDISCRETETYPE 2376 // VariableType
#define UA_NS0ID_PROGRAMTRANSITIONEVENTTYPE 2378 // ObjectType
#define UA_NS0ID_PROGRAMDIAGNOSTICTYPE 2380 // VariableType
#define UA_NS0ID_PROGRAMSTATEMACHINETYPE 2391 // ObjectType
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_MAXBROWSECONTINUATIONPOINTS 2735 //␣
→Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_MAXQUERYCONTINUATIONPOINTS 2736 //␣
→Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_MAXHISTORYCONTINUATIONPOINTS 2737 //␣
→Variable
#define UA_NS0ID_SEMANTICCHANGEEVENTTYPE 2738 // ObjectType
#define UA_NS0ID_AUDITURLMISMATCHEVENTTYPE 2748 // ObjectType
#define UA_NS0ID_STATEVARIABLETYPE 2755 // VariableType
#define UA_NS0ID_FINITESTATEVARIABLETYPE 2760 // VariableType
#define UA_NS0ID_TRANSITIONVARIABLETYPE 2762 // VariableType
#define UA_NS0ID_FINITETRANSITIONVARIABLETYPE 2767 // VariableType
#define UA_NS0ID_FINITESTATEMACHINETYPE 2771 // ObjectType
#define UA_NS0ID_CONDITIONTYPE 2782 // ObjectType
#define UA_NS0ID_REFRESHSTARTEVENTTYPE 2787 // ObjectType
#define UA_NS0ID_REFRESHENDEVENTTYPE 2788 // ObjectType
#define UA_NS0ID_REFRESHREQUIREDEVENTTYPE 2789 // ObjectType
#define UA_NS0ID_AUDITCONDITIONEVENTTYPE 2790 // ObjectType
#define UA_NS0ID_AUDITCONDITIONENABLEEVENTTYPE 2803 // ObjectType
#define UA_NS0ID_AUDITCONDITIONCOMMENTEVENTTYPE 2829 // ObjectType
#define UA_NS0ID_DIALOGCONDITIONTYPE 2830 // ObjectType
#define UA_NS0ID_ACKNOWLEDGEABLECONDITIONTYPE 2881 // ObjectType
#define UA_NS0ID_ALARMCONDITIONTYPE 2915 // ObjectType
#define UA_NS0ID_SHELVEDSTATEMACHINETYPE 2929 // ObjectType
#define UA_NS0ID_LIMITALARMTYPE 2955 // ObjectType
#define UA_NS0ID_SERVER_SERVERSTATUS_SECONDSTILLSHUTDOWN 2992 // Variable
#define UA_NS0ID_SERVER_SERVERSTATUS_SHUTDOWNREASON 2993 // Variable
#define UA_NS0ID_SERVER_AUDITING 2994 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_MODELLINGRULES 2996 // Object
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_AGGREGATEFUNCTIONS 2997 // Object
#define UA_NS0ID_AUDITHISTORYEVENTUPDATEEVENTTYPE 2999 // ObjectType
#define UA_NS0ID_AUDITHISTORYVALUEUPDATEEVENTTYPE 3006 // ObjectType
#define UA_NS0ID_AUDITHISTORYDELETEEVENTTYPE 3012 // ObjectType
#define UA_NS0ID_AUDITHISTORYRAWMODIFYDELETEEVENTTYPE 3014 // ObjectType
#define UA_NS0ID_AUDITHISTORYATTIMEDELETEEVENTTYPE 3019 // ObjectType
#define UA_NS0ID_AUDITHISTORYEVENTDELETEEVENTTYPE 3022 // ObjectType
#define UA_NS0ID_EVENTQUEUEOVERFLOWEVENTTYPE 3035 // ObjectType
#define UA_NS0ID_EVENTTYPESFOLDER 3048 // Object
#define UA_NS0ID_BUILDINFOTYPE 3051 // VariableType
#define UA_NS0ID_DEFAULTBINARY 3062 // Object
#define UA_NS0ID_DEFAULTXML 3063 // Object
#define UA_NS0ID_ALWAYSGENERATESEVENT 3065 // ReferenceType
#define UA_NS0ID_ICON 3067 // Variable
#define UA_NS0ID_NODEVERSION 3068 // Variable
#define UA_NS0ID_LOCALTIME 3069 // Variable
```

Chapter 10. Standard-Defined Constants

```
#define UA_NS0ID_ALLOWNULLS 3070 // Variable
#define UA_NS0ID_ENUMVALUES 3071 // Variable
#define UA_NS0ID_INPUTARGUMENTS 3072 // Variable
#define UA_NS0ID_OUTPUTARGUMENTS 3073 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_SOFTWARECERTIFICATES 3704 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SERVERDIAGNOSTICSSUMMARY_
↪REJECTEDSESSIONCOUNT 3705 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SESSIONSDIAGNOSTICSSUMMARY 3706 // Object
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SESSIONSDIAGNOSTICSSUMMARY_
↪SESSIONDIAGNOSTICSARRAY 3707 // Variable
#define UA_NS0ID_SERVER_SERVERDIAGNOSTICS_SESSIONSDIAGNOSTICSSUMMARY_
↪SESSIONSECURITYDIAGNOSTICSARRAY 3708 // Variable
#define UA_NS0ID_SERVER_SERVERREDUNDANCY_REDUNDANCYSUPPORT 3709 // Variable
#define UA_NS0ID_PROGRAMTRANSITIONAUDITEVENTTYPE 3806 // ObjectType
#define UA_NS0ID_ADDCOMMENTMETHODTYPE 3863 // Method
#define UA_NS0ID_TIMEDSHELVEMETHODTYPE 6102 // Method
#define UA_NS0ID_ENUMVALUETYPE 7594 // DataType
#define UA_NS0ID_MESSAGESECURITYMODE_ENUMSTRINGS 7595 // Variable
#define UA_NS0ID_BROWSEDIRECTION_ENUMSTRINGS 7603 // Variable
#define UA_NS0ID_FILTEROPERATOR_ENUMSTRINGS 7605 // Variable
#define UA_NS0ID_TIMESTAMPSTORETURN_ENUMSTRINGS 7606 // Variable
#define UA_NS0ID_MONITORINGMODE_ENUMSTRINGS 7608 // Variable
#define UA_NS0ID_DATACHANGETRIGGER_ENUMSTRINGS 7609 // Variable
#define UA_NS0ID_REDUNDANCYSUPPORT_ENUMSTRINGS 7611 // Variable
#define UA_NS0ID_SERVERSTATE_ENUMSTRINGS 7612 // Variable
#define UA_NS0ID_EXCEPTIONDEVIATIONFORMAT_ENUMSTRINGS 7614 // Variable
#define UA_NS0ID_TIMEZONEDATATYPE 8912 // DataType
#define UA_NS0ID_AUDITCONDITIONRESPONDEVENTTYPE 8927 // ObjectType
#define UA_NS0ID_AUDITCONDITIONACKNOWLEDGEEVENTTYPE 8944 // ObjectType
#define UA_NS0ID_AUDITCONDITIONCONFIRMEVENTTYPE 8961 // ObjectType
#define UA_NS0ID_TWOSTATEVARIABLETYPE 8995 // VariableType
#define UA_NS0ID_CONDITIONVARIABLETYPE 9002 // VariableType
#define UA_NS0ID_HASTRUESUBSTATE 9004 // ReferenceType
#define UA_NS0ID_HASFALSESUBSTATE 9005 // ReferenceType
#define UA_NS0ID_HASCONDITION 9006 // ReferenceType
#define UA_NS0ID_CONDITIONREFRESHMETHODTYPE 9007 // Method
#define UA_NS0ID_DIALOGRESPONSEMETHODTYPE 9031 // Method
#define UA_NS0ID_EXCLUSIVELIMITSTATEMACHINETYPE 9318 // ObjectType
#define UA_NS0ID_EXCLUSIVELIMITALARMTYPE 9341 // ObjectType
#define UA_NS0ID_EXCLUSIVELEVELALARMTYPE 9482 // ObjectType
#define UA_NS0ID_EXCLUSIVERATEOFCHANGEALARMTYPE 9623 // ObjectType
#define UA_NS0ID_EXCLUSIVEDEVIATIONALARMTYPE 9764 // ObjectType
#define UA_NS0ID_NONEXCLUSIVELIMITALARMTYPE 9906 // ObjectType
#define UA_NS0ID_NONEXCLUSIVELEVELALARMTYPE 10060 // ObjectType
#define UA_NS0ID_NONEXCLUSIVERATEOFCHANGEALARMTYPE 10214 // ObjectType
#define UA_NS0ID_NONEXCLUSIVEDEVIATIONALARMTYPE 10368 // ObjectType
#define UA_NS0ID_DISCRETEALARMTYPE 10523 // ObjectType
#define UA_NS0ID_OFFNORMALALARMTYPE 10637 // ObjectType
#define UA_NS0ID_TRIPALARMTYPE 10751 // ObjectType
#define UA_NS0ID_AUDITCONDITIONSHELVINGEVENTTYPE 11093 // ObjectType
#define UA_NS0ID_BASECONDITIONCLASSTYPE 11163 // ObjectType
#define UA_NS0ID_PROCESSCONDITIONCLASSTYPE 11164 // ObjectType
#define UA_NS0ID_MAINTENANCECONDITIONCLASSTYPE 11165 // ObjectType
#define UA_NS0ID_SYSTEMCONDITIONCLASSTYPE 11166 // ObjectType
#define UA_NS0ID_AGGREGATECONFIGURATIONTYPE 11187 // ObjectType
#define UA_NS0ID_HISTORYSERVERCAPABILITIES 11192 // Object
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_ACCESSHISTORYDATACAPABILITY 11193 //␣
↪Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_INSERTDATACAPABILITY 11196 // Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_REPLACEDATACAPABILITY 11197 // Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_UPDATEDATACAPABILITY 11198 // Variable
```

```
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_DELETERAWCAPABILITY 11199 // Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_DELETEATTIMECAPABILITY 11200 // Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_AGGREGATEFUNCTIONS 11201 // Object
#define UA_NS0ID_HACONFIGURATION 11202 // Object
#define UA_NS0ID_HACONFIGURATION_AGGREGATECONFIGURATION 11203 // Object
#define UA_NS0ID_HACONFIGURATION_AGGREGATECONFIGURATION_TREATUNCERTAINASBAD 11204 /
↪/ Variable
#define UA_NS0ID_HACONFIGURATION_AGGREGATECONFIGURATION_PERCENTDATABAD 11205 //␣
↪Variable
#define UA_NS0ID_HACONFIGURATION_AGGREGATECONFIGURATION_PERCENTDATAGOOD 11206 //␣
↪Variable
#define UA_NS0ID_HACONFIGURATION_AGGREGATECONFIGURATION_USESLOPEDEXTRAPOLATION␣
↪11207 // Variable
#define UA_NS0ID_HACONFIGURATION_STEPPED 11208 // Variable
#define UA_NS0ID_HACONFIGURATION_DEFINITION 11209 // Variable
#define UA_NS0ID_HACONFIGURATION_MAXTIMEINTERVAL 11210 // Variable
#define UA_NS0ID_HACONFIGURATION_MINTIMEINTERVAL 11211 // Variable
#define UA_NS0ID_HACONFIGURATION_EXCEPTIONDEVIATION 11212 // Variable
#define UA_NS0ID_HACONFIGURATION_EXCEPTIONDEVIATIONFORMAT 11213 // Variable
#define UA_NS0ID_ANNOTATIONS 11214 // Variable
#define UA_NS0ID_HISTORICALEVENTFILTER 11215 // Variable
#define UA_NS0ID_MODIFICATIONINFO 11216 // DataType
#define UA_NS0ID_HISTORYMODIFIEDDATA 11217 // DataType
#define UA_NS0ID_MODIFICATIONINFO_ENCODING_DEFAULTXML 11218 // Object
#define UA_NS0ID_HISTORYMODIFIEDDATA_ENCODING_DEFAULTXML 11219 // Object
#define UA_NS0ID_MODIFICATIONINFO_ENCODING_DEFAULTBINARY 11226 // Object
#define UA_NS0ID_HISTORYMODIFIEDDATA_ENCODING_DEFAULTBINARY 11227 // Object
#define UA_NS0ID_HISTORYUPDATETYPE 11234 // DataType
#define UA_NS0ID_MULTISTATEVALUEDISCRETETYPE 11238 // VariableType
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_ACCESSHISTORYEVENTSCAPABILITY 11242 //␣
↪Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_MAXRETURNDATAVALUES 11273 // Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_MAXRETURNEVENTVALUES 11274 // Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_INSERTANNOTATIONCAPABILITY 11275 //␣
↪Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_INSERTEVENTCAPABILITY 11281 // Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_REPLACEEVENTCAPABILITY 11282 // Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_UPDATEEVENTCAPABILITY 11283 // Variable
#define UA_NS0ID_AGGREGATEFUNCTION_TIMEAVERAGE2 11285 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_MINIMUM2 11286 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_MAXIMUM2 11287 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_RANGE2 11288 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_WORSTQUALITY2 11292 // Object
#define UA_NS0ID_PERFORMUPDATETYPE 11293 // DataType
#define UA_NS0ID_UPDATESTRUCTUREDATADETAILS 11295 // DataType
#define UA_NS0ID_UPDATESTRUCTUREDATADETAILS_ENCODING_DEFAULTXML 11296 // Object
#define UA_NS0ID_UPDATESTRUCTUREDATADETAILS_ENCODING_DEFAULTBINARY 11300 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_TOTAL2 11304 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_MINIMUMACTUALTIME2 11305 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_MAXIMUMACTUALTIME2 11306 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_DURATIONINSTATEZERO 11307 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_DURATIONINSTATENONZERO 11308 // Object
#define UA_NS0ID_SERVER_SERVERREDUNDANCY_CURRENTSERVERID 11312 // Variable
#define UA_NS0ID_SERVER_SERVERREDUNDANCY_REDUNDANTSERVERARRAY 11313 // Variable
#define UA_NS0ID_SERVER_SERVERREDUNDANCY_SERVERURIARRAY 11314 // Variable
#define UA_NS0ID_AGGREGATEFUNCTION_STANDARDDEVIATIONSAMPLE 11426 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_STANDARDDEVIATIONPOPULATION 11427 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_VARIANCESAMPLE 11428 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_VARIANCEPOPULATION 11429 // Object
#define UA_NS0ID_ENUMSTRINGS 11432 // Variable
#define UA_NS0ID_VALUEASTEXT 11433 // Variable
```

```
#define UA_NS0ID_PROGRESSEVENTTYPE 11436 // ObjectType
#define UA_NS0ID_SYSTEMSTATUSCHANGEEVENTTYPE 11446 // ObjectType
#define UA_NS0ID_OPTIONSETTYPE 11487 // VariableType
#define UA_NS0ID_SERVER_GETMONITOREDITEMS 11492 // Method
#define UA_NS0ID_SERVER_GETMONITOREDITEMS_INPUTARGUMENTS 11493 // Variable
#define UA_NS0ID_SERVER_GETMONITOREDITEMS_OUTPUTARGUMENTS 11494 // Variable
#define UA_NS0ID_GETMONITOREDITEMSMETHODTYPE 11495 // Method
#define UA_NS0ID_MAXSTRINGLENGTH 11498 // Variable
#define UA_NS0ID_HISTORYSERVERCAPABILITIES_DELETEEVENTCAPABILITY 11502 // Variable
#define UA_NS0ID_HACONFIGURATION_STARTOFARCHIVE 11503 // Variable
#define UA_NS0ID_HACONFIGURATION_STARTOFONLINEARCHIVE 11504 // Variable
#define UA_NS0ID_AGGREGATEFUNCTION_STARTBOUND 11505 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_ENDBOUND 11506 // Object
#define UA_NS0ID_AGGREGATEFUNCTION_DELTABOUNDS 11507 // Object
#define UA_NS0ID_MODELLINGRULE_OPTIONALPLACEHOLDER 11508 // Object
#define UA_NS0ID_MODELLINGRULE_OPTIONALPLACEHOLDER_NAMINGRULE 11509 // Variable
#define UA_NS0ID_MODELLINGRULE_MANDATORYPLACEHOLDER 11510 // Object
#define UA_NS0ID_MODELLINGRULE_MANDATORYPLACEHOLDER_NAMINGRULE 11511 // Variable
#define UA_NS0ID_MAXARRAYLENGTH 11512 // Variable
#define UA_NS0ID_ENGINEERINGUNITS 11513 // Variable
#define UA_NS0ID_OPERATIONLIMITSTYPE 11564 // ObjectType
#define UA_NS0ID_FILETYPE 11575 // ObjectType
#define UA_NS0ID_ADDRESSSPACEFILETYPE 11595 // ObjectType
#define UA_NS0ID_NAMESPACEMETADATATYPE 11616 // ObjectType
#define UA_NS0ID_NAMESPACESTYPE 11645 // ObjectType
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_MAXARRAYLENGTH 11702 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_MAXSTRINGLENGTH 11703 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS 11704 // Object
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_MAXNODESPERREAD 11705 //
→ Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_MAXNODESPERWRITE 11707 /
→/ Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_MAXNODESPERMETHODCALL␣
→11709 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_MAXNODESPERBROWSE 11710␣
→// Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_
→MAXNODESPERREGISTERNODES 11711 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_
→MAXNODESPERTRANSLATEBROWSEPATHSTONODEIDS 11712 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_
→MAXNODESPERNODEMANAGEMENT 11713 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_
→MAXMONITOREDITEMSPERCALL 11714 // Variable
#define UA_NS0ID_SERVER_NAMESPACES 11715 // Object
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE 11716 // Object
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_SIZE 11717 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_OPENCOUNT 11720 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_OPEN 11721 // Method
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_OPEN_INPUTARGUMENTS 11722 //␣
→Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_OPEN_OUTPUTARGUMENTS 11723 //␣
→Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_CLOSE 11724 // Method
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_CLOSE_INPUTARGUMENTS 11725 //␣
→Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_READ 11726 // Method
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_READ_INPUTARGUMENTS 11727 //␣
→Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_READ_OUTPUTARGUMENTS 11728 //␣
→Variable
```

```
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_WRITE 11729 // Method
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_WRITE_INPUTARGUMENTS 11730 //␣
↪Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_GETPOSITION 11731 // Method
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_GETPOSITION_INPUTARGUMENTS␣
↪11732 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_GETPOSITION_OUTPUTARGUMENTS␣
↪11733 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_SETPOSITION 11734 // Method
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_SETPOSITION_INPUTARGUMENTS␣
↪11735 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_EXPORTNAMESPACE 11736 // Method
#define UA_NS0ID_BITFIELDMASKDATATYPE 11737 // DataType
#define UA_NS0ID_OPENMETHODTYPE 11738 // Method
#define UA_NS0ID_CLOSEMETHODTYPE 11741 // Method
#define UA_NS0ID_READMETHODTYPE 11743 // Method
#define UA_NS0ID_WRITEMETHODTYPE 11746 // Method
#define UA_NS0ID_GETPOSITIONMETHODTYPE 11748 // Method
#define UA_NS0ID_SETPOSITIONMETHODTYPE 11751 // Method
#define UA_NS0ID_SYSTEMOFFNORMALALARMTYPE 11753 // ObjectType
#define UA_NS0ID_AUDITPROGRAMTRANSITIONEVENTTYPE 11856 // ObjectType
#define UA_NS0ID_HACONFIGURATION_AGGREGATEFUNCTIONS 11877 // Object
#define UA_NS0ID_NODECLASS_ENUMVALUES 11878 // Variable
#define UA_NS0ID_INSTANCENODE 11879 // DataType
#define UA_NS0ID_TYPENODE 11880 // DataType
#define UA_NS0ID_NODEATTRIBUTESMASK_ENUMVALUES 11881 // Variable
#define UA_NS0ID_ATTRIBUTEWRITEMASK_ENUMVALUES 11882 // Variable
#define UA_NS0ID_BROWSERESULTMASK_ENUMVALUES 11883 // Variable
#define UA_NS0ID_INSTANCENODE_ENCODING_DEFAULTXML 11887 // Object
#define UA_NS0ID_TYPENODE_ENCODING_DEFAULTXML 11888 // Object
#define UA_NS0ID_INSTANCENODE_ENCODING_DEFAULTBINARY 11889 // Object
#define UA_NS0ID_TYPENODE_ENCODING_DEFAULTBINARY 11890 // Object
#define UA_NS0ID_OPENFILEMODE 11939 // DataType
#define UA_NS0ID_OPENFILEMODE_ENUMVALUES 11940 // Variable
#define UA_NS0ID_MODELCHANGESTRUCTUREVERBMASK 11941 // DataType
#define UA_NS0ID_MODELCHANGESTRUCTUREVERBMASK_ENUMVALUES 11942 // Variable
#define UA_NS0ID_ENDPOINTURLLISTDATATYPE 11943 // DataType
#define UA_NS0ID_NETWORKGROUPDATATYPE 11944 // DataType
#define UA_NS0ID_NONTRANSPARENTNETWORKREDUNDANCYTYPE 11945 // ObjectType
#define UA_NS0ID_ARRAYITEMTYPE 12021 // VariableType
#define UA_NS0ID_YARRAYITEMTYPE 12029 // VariableType
#define UA_NS0ID_XYARRAYITEMTYPE 12038 // VariableType
#define UA_NS0ID_IMAGEITEMTYPE 12047 // VariableType
#define UA_NS0ID_CUBEITEMTYPE 12057 // VariableType
#define UA_NS0ID_NDIMENSIONARRAYITEMTYPE 12068 // VariableType
#define UA_NS0ID_AXISSCALEENUMERATION 12077 // DataType
#define UA_NS0ID_AXISSCALEENUMERATION_ENUMSTRINGS 12078 // Variable
#define UA_NS0ID_AXISINFORMATION 12079 // DataType
#define UA_NS0ID_XVTYPE 12080 // DataType
#define UA_NS0ID_AXISINFORMATION_ENCODING_DEFAULTXML 12081 // Object
#define UA_NS0ID_AXISINFORMATION_ENCODING_DEFAULTBINARY 12089 // Object
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_
↪MAXNODESPERHISTORYREADDATA 12165 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_
↪MAXNODESPERHISTORYREADEVENTS 12166 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_
↪MAXNODESPERHISTORYUPDATEDATA 12167 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_OPERATIONLIMITS_
↪MAXNODESPERHISTORYUPDATEEVENTS 12168 // Variable
#define UA_NS0ID_VIEWVERSION 12170 // Variable
#define UA_NS0ID_COMPLEXNUMBERTYPE 12171 // DataType
```

```
#define UA_NS0ID_DOUBLECOMPLEXNUMBERTYPE 12172 // DataType
#define UA_NS0ID_SERVERONNETWORK 12189 // DataType
#define UA_NS0ID_FINDSERVERSONNETWORKREQUEST 12190 // DataType
#define UA_NS0ID_FINDSERVERSONNETWORKRESPONSE 12191 // DataType
#define UA_NS0ID_REGISTERSERVER2REQUEST 12193 // DataType
#define UA_NS0ID_REGISTERSERVER2RESPONSE 12194 // DataType
#define UA_NS0ID_SERVERONNETWORK_ENCODING_DEFAULTXML 12195 // Object
#define UA_NS0ID_FINDSERVERSONNETWORKREQUEST_ENCODING_DEFAULTXML 12196 // Object
#define UA_NS0ID_FINDSERVERSONNETWORKRESPONSE_ENCODING_DEFAULTXML 12197 // Object
#define UA_NS0ID_REGISTERSERVER2REQUEST_ENCODING_DEFAULTXML 12199 // Object
#define UA_NS0ID_REGISTERSERVER2RESPONSE_ENCODING_DEFAULTXML 12200 // Object
#define UA_NS0ID_SERVERONNETWORK_ENCODING_DEFAULTBINARY 12207 // Object
#define UA_NS0ID_FINDSERVERSONNETWORKREQUEST_ENCODING_DEFAULTBINARY 12208 // Object
#define UA_NS0ID_FINDSERVERSONNETWORKRESPONSE_ENCODING_DEFAULTBINARY 12209 //␣
→Object
#define UA_NS0ID_REGISTERSERVER2REQUEST_ENCODING_DEFAULTBINARY 12211 // Object
#define UA_NS0ID_REGISTERSERVER2RESPONSE_ENCODING_DEFAULTBINARY 12212 // Object
#define UA_NS0ID_OPENWITHMASKSMETHODTYPE 12513 // Method
#define UA_NS0ID_CLOSEANDUPDATEMETHODTYPE 12516 // Method
#define UA_NS0ID_ADDCERTIFICATEMETHODTYPE 12518 // Method
#define UA_NS0ID_REMOVECERTIFICATEMETHODTYPE 12520 // Method
#define UA_NS0ID_TRUSTLISTTYPE 12522 // ObjectType
#define UA_NS0ID_TRUSTLISTMASKS 12552 // DataType
#define UA_NS0ID_TRUSTLISTMASKS_ENUMVALUES 12553 // Variable
#define UA_NS0ID_TRUSTLISTDATATYPE 12554 // DataType
#define UA_NS0ID_CERTIFICATEGROUPTYPE 12555 // ObjectType
#define UA_NS0ID_CERTIFICATETYPE 12556 // ObjectType
#define UA_NS0ID_APPLICATIONCERTIFICATETYPE 12557 // ObjectType
#define UA_NS0ID_HTTPSCERTIFICATETYPE 12558 // ObjectType
#define UA_NS0ID_RSAMINAPPLICATIONCERTIFICATETYPE 12559 // ObjectType
#define UA_NS0ID_RSASHA256APPLICATIONCERTIFICATETYPE 12560 // ObjectType
#define UA_NS0ID_TRUSTLISTUPDATEDAUDITEVENTTYPE 12561 // ObjectType
#define UA_NS0ID_UPDATECERTIFICATEMETHODTYPE 12578 // Method
#define UA_NS0ID_SERVERCONFIGURATIONTYPE 12581 // ObjectType
#define UA_NS0ID_CERTIFICATEUPDATEDAUDITEVENTTYPE 12620 // ObjectType
#define UA_NS0ID_SERVERCONFIGURATION 12637 // Object
#define UA_NS0ID_SERVERCONFIGURATION_SUPPORTEDPRIVATEKEYFORMATS 12639 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_MAXTRUSTLISTSIZE 12640 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_MULTICASTDNSENABLED 12641 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST 12642 // Object
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_SIZE 12643 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_OPENCOUNT 12646 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_OPEN 12647 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_OPEN_INPUTARGUMENTS 12648 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_OPEN_OUTPUTARGUMENTS 12649 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_CLOSE 12650 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_CLOSE_INPUTARGUMENTS 12651 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_READ 12652 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_READ_INPUTARGUMENTS 12653 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_READ_OUTPUTARGUMENTS 12654 // Variable
```

```
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_WRITE 12655 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_WRITE_INPUTARGUMENTS 12656 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_GETPOSITION 12657 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_GETPOSITION_INPUTARGUMENTS 12658 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_GETPOSITION_OUTPUTARGUMENTS 12659 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_SETPOSITION 12660 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_SETPOSITION_INPUTARGUMENTS 12661 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_LASTUPDATETIME 12662 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_OPENWITHMASKS 12663 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_OPENWITHMASKS_INPUTARGUMENTS 12664 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_OPENWITHMASKS_OUTPUTARGUMENTS 12665 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_CLOSEANDUPDATE 12666 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_CLOSEANDUPDATE_OUTPUTARGUMENTS 12667 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_ADDCERTIFICATE 12668 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_ADDCERTIFICATE_INPUTARGUMENTS 12669 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_REMOVECERTIFICATE 12670 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_REMOVECERTIFICATE_INPUTARGUMENTS 12671 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_WRITABLE 12696 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_USERWRITABLE 12697 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_SERVERCAPABILITIES 12710 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CREATESIGNINGREQUEST 12737 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CREATESIGNINGREQUEST_INPUTARGUMENTS 12738 //␣
→Variable
#define UA_NS0ID_SERVERCONFIGURATION_CREATESIGNINGREQUEST_OUTPUTARGUMENTS 12739 //␣
→Variable
#define UA_NS0ID_SERVERCONFIGURATION_APPLYCHANGES 12740 // Method
#define UA_NS0ID_CREATESIGNINGREQUESTMETHODTYPE 12741 // Method
#define UA_NS0ID_OPTIONSETVALUES 12745 // Variable
#define UA_NS0ID_SERVER_SETSUBSCRIPTIONDURABLE 12749 // Method
#define UA_NS0ID_SERVER_SETSUBSCRIPTIONDURABLE_INPUTARGUMENTS 12750 // Variable
#define UA_NS0ID_SERVER_SETSUBSCRIPTIONDURABLE_OUTPUTARGUMENTS 12751 // Variable
#define UA_NS0ID_SETSUBSCRIPTIONDURABLEMETHODTYPE 12752 // Method
#define UA_NS0ID_OPTIONSET 12755 // DataType
#define UA_NS0ID_UNION 12756 // DataType
#define UA_NS0ID_OPTIONSET_ENCODING_DEFAULTXML 12757 // Object
#define UA_NS0ID_UNION_ENCODING_DEFAULTXML 12758 // Object
#define UA_NS0ID_OPTIONSET_ENCODING_DEFAULTBINARY 12765 // Object
#define UA_NS0ID_UNION_ENCODING_DEFAULTBINARY 12766 // Object
#define UA_NS0ID_GETREJECTEDLISTMETHODTYPE 12773 // Method
#define UA_NS0ID_SERVERCONFIGURATION_GETREJECTEDLIST 12777 // Method
#define UA_NS0ID_SERVERCONFIGURATION_GETREJECTEDLIST_OUTPUTARGUMENTS 12778 //␣
→Variable
#define UA_NS0ID_SERVER_RESENDDATA 12873 // Method
#define UA_NS0ID_SERVER_RESENDDATA_INPUTARGUMENTS 12874 // Variable
```

```
#define UA_NS0ID_RESENDDATAMETHODTYPE 12875 // Method
#define UA_NS0ID_NORMALIZEDSTRING 12877 // DataType
#define UA_NS0ID_DECIMALSTRING 12878 // DataType
#define UA_NS0ID_DURATIONSTRING 12879 // DataType
#define UA_NS0ID_TIMESTRING 12880 // DataType
#define UA_NS0ID_DATESTRING 12881 // DataType
#define UA_NS0ID_SERVER_ESTIMATEDRETURNTIME 12885 // Variable
#define UA_NS0ID_SERVER_REQUESTSERVERSTATECHANGE 12886 // Method
#define UA_NS0ID_SERVER_REQUESTSERVERSTATECHANGE_INPUTARGUMENTS 12887 // Variable
#define UA_NS0ID_REQUESTSERVERSTATECHANGEMETHODTYPE 12888 // Method
#define UA_NS0ID_DISCOVERYCONFIGURATION 12890 // DataType
#define UA_NS0ID_MDNSDISCOVERYCONFIGURATION 12891 // DataType
#define UA_NS0ID_DISCOVERYCONFIGURATION_ENCODING_DEFAULTXML 12892 // Object
#define UA_NS0ID_MDNSDISCOVERYCONFIGURATION_ENCODING_DEFAULTXML 12893 // Object
#define UA_NS0ID_DISCOVERYCONFIGURATION_ENCODING_DEFAULTBINARY 12900 // Object
#define UA_NS0ID_MDNSDISCOVERYCONFIGURATION_ENCODING_DEFAULTBINARY 12901 // Object
#define UA_NS0ID_MAXBYTESTRINGLENGTH 12908 // Variable
#define UA_NS0ID_SERVER_SERVERCAPABILITIES_MAXBYTESTRINGLENGTH 12911 // Variable
#define UA_NS0ID_CONDITIONREFRESH2METHODTYPE 12914 // Method
#define UA_NS0ID_CERTIFICATEEXPIRATIONALARMTYPE 13225 // ObjectType
#define UA_NS0ID_CREATEDIRECTORYMETHODTYPE 13342 // Method
#define UA_NS0ID_CREATEFILEMETHODTYPE 13345 // Method
#define UA_NS0ID_DELETEFILEMETHODTYPE 13348 // Method
#define UA_NS0ID_MOVEORCOPYMETHODTYPE 13350 // Method
#define UA_NS0ID_FILEDIRECTORYTYPE 13353 // ObjectType
#define UA_NS0ID_SERVER_NAMESPACES_ADDRESSSPACEFILE_MIMETYPE 13402 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_UPDATECERTIFICATE 13737 // Method
#define UA_NS0ID_SERVERCONFIGURATION_UPDATECERTIFICATE_INPUTARGUMENTS 13738 //␣
→Variable
#define UA_NS0ID_SERVERCONFIGURATION_UPDATECERTIFICATE_OUTPUTARGUMENTS 13739 //␣
→Variable
#define UA_NS0ID_CERTIFICATEGROUPFOLDERTYPE 13813 // ObjectType
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS 14053 // Object
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP 14088 //␣
→Object
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST␣
→14089 // Object
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→SIZE 14090 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→WRITABLE 14091 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→USERWRITABLE 14092 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→OPENCOUNT 14093 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→MIMETYPE 14094 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→OPEN 14095 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→OPEN_INPUTARGUMENTS 14096 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→OPEN_OUTPUTARGUMENTS 14097 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→CLOSE 14098 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→CLOSE_INPUTARGUMENTS 14099 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→READ 14100 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
→READ_INPUTARGUMENTS 14101 // Variable
```

```
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪READ_OUTPUTARGUMENTS 14102 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪WRITE 14103 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪WRITE_INPUTARGUMENTS 14104 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪GETPOSITION 14105 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪GETPOSITION_INPUTARGUMENTS 14106 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪GETPOSITION_OUTPUTARGUMENTS 14107 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪SETPOSITION 14108 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪SETPOSITION_INPUTARGUMENTS 14109 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪LASTUPDATETIME 14110 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪OPENWITHMASKS 14111 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪OPENWITHMASKS_INPUTARGUMENTS 14112 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪OPENWITHMASKS_OUTPUTARGUMENTS 14113 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪CLOSEANDUPDATE 14114 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪CLOSEANDUPDATE_INPUTARGUMENTS 14115 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪CLOSEANDUPDATE_OUTPUTARGUMENTS 14116 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪ADDCERTIFICATE 14117 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪ADDCERTIFICATE_INPUTARGUMENTS 14118 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪REMOVECERTIFICATE 14119 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_TRUSTLIST_
↪REMOVECERTIFICATE_INPUTARGUMENTS 14120 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTHTTPSGROUP_
↪CERTIFICATETYPES 14121 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP 14122␣
↪// Object
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
↪TRUSTLIST 14123 // Object
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
↪TRUSTLIST_SIZE 14124 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
↪TRUSTLIST_WRITABLE 14125 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
↪TRUSTLIST_USERWRITABLE 14126 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
↪TRUSTLIST_OPENCOUNT 14127 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
↪TRUSTLIST_MIMETYPE 14128 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
↪TRUSTLIST_OPEN 14129 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
↪TRUSTLIST_OPEN_INPUTARGUMENTS 14130 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
↪TRUSTLIST_OPEN_OUTPUTARGUMENTS 14131 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
↪TRUSTLIST_CLOSE 14132 // Method
```

```
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_CLOSE_INPUTARGUMENTS 14133 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_READ 14134 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_READ_INPUTARGUMENTS 14135 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_READ_OUTPUTARGUMENTS 14136 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_WRITE 14137 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_WRITE_INPUTARGUMENTS 14138 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_GETPOSITION 14139 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_GETPOSITION_INPUTARGUMENTS 14140 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_GETPOSITION_OUTPUTARGUMENTS 14141 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_SETPOSITION 14142 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_SETPOSITION_INPUTARGUMENTS 14143 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_LASTUPDATETIME 14144 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_OPENWITHMASKS 14145 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_OPENWITHMASKS_INPUTARGUMENTS 14146 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_OPENWITHMASKS_OUTPUTARGUMENTS 14147 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_CLOSEANDUPDATE 14148 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_CLOSEANDUPDATE_INPUTARGUMENTS 14149 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_CLOSEANDUPDATE_OUTPUTARGUMENTS 14150 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_ADDCERTIFICATE 14151 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_ADDCERTIFICATE_INPUTARGUMENTS 14152 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_REMOVECERTIFICATE 14153 // Method
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→TRUSTLIST_REMOVECERTIFICATE_INPUTARGUMENTS 14154 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTUSERTOKENGROUP_
→CERTIFICATETYPES 14155 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→14156 // Object
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_WRITABLE 14157 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_USERWRITABLE 14158 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_MIMETYPE 14159 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→TRUSTLIST_CLOSEANDUPDATE_INPUTARGUMENTS 14160 // Variable
#define UA_NS0ID_SERVERCONFIGURATION_CERTIFICATEGROUPS_DEFAULTAPPLICATIONGROUP_
→CERTIFICATETYPES 14161 // Variable
#define UA_NS0ID_SERVER_SERVERREDUNDANCY_SERVERNETWORKGROUPS 14415 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI 15182 // Object
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEURI 15183 // Variable
```

```
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEVERSION 15184 //␣
→Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEPUBLICATIONDATE␣
→15185 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_ISNAMESPACESUBSET 15186 //␣
→Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_STATICNODEIDTYPES 15187 //␣
→Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_STATICNUMERICNODEIDRANGE␣
→15188 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_STATICSTRINGNODEIDPATTERN␣
→15189 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE 15190 // Object
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_SIZE 15191 //␣
→Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_WRITABLE 15192 /
→/ Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_USERWRITABLE␣
→15193 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_OPENCOUNT 15194␣
→// Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_MIMETYPE 15195 /
→/ Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_OPEN 15196 //␣
→Method
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_OPEN_
→INPUTARGUMENTS 15197 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_OPEN_
→OUTPUTARGUMENTS 15198 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_CLOSE 15199 //␣
→Method
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_CLOSE_
→INPUTARGUMENTS 15200 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_READ 15201 //␣
→Method
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_READ_
→INPUTARGUMENTS 15202 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_READ_
→OUTPUTARGUMENTS 15203 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_WRITE 15204 //␣
→Method
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_WRITE_
→INPUTARGUMENTS 15205 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_GETPOSITION␣
→15206 // Method
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_GETPOSITION_
→INPUTARGUMENTS 15207 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_GETPOSITION_
→OUTPUTARGUMENTS 15208 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_SETPOSITION␣
→15209 // Method
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_SETPOSITION_
→INPUTARGUMENTS 15210 // Variable
#define UA_NS0ID_SERVER_NAMESPACES_OPCUANAMESPACEURI_NAMESPACEFILE_EXPORTNAMESPACE␣
→15211 // Method
#define UA_NS0ID_HASMODELPARENT 50 // ReferenceType


#define UA_VALUERANK_SCALAR_OR_ONE_DIMENSION   -3
#define UA_VALUERANK_ANY                       -2
#define UA_VALUERANK_SCALAR                    -1
#define UA_VALUERANK_ONE_OR_MORE_DIMENSIONS     0
```

```
#define UA_VALUERANK_ONE_DIMENSION           1
#define UA_VALUERANK_TWO_DIMENSIONS          2
#define UA_VALUERANK_THREE_DIMENSIONS        3
```

XML Nodeset Compiler

When writing an application, it is more comfortable to create information models using some GUI tools. Most tools can export data according the OPC UA Nodeset XML schema. open62541 contains a python based nodeset compiler that can transform these information model definitions into a working server.

Note that the nodeset compiler you can find in the *tools/nodeset_compiler* subfolder is *not* an XML transformation tool but a compiler. That means that it will create an internal representation when parsing the XML files and attempt to understand and verify the correctness of this representation in order to generate C Code.

11.1 Getting started

We take the following information model snippet as the starting point of the following tutorial:

```xml
<UANodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:uax="http://opcfoundation.org/UA/2008/02/Types.xsd"
           xmlns="http://opcfoundation.org/UA/2011/03/UANodeSet.xsd"
           xmlns:s1="http://yourorganisation.org/example_nodeset/"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <NamespaceUris>
        <Uri>http://yourorganisation.org/example_nodeset/</Uri>
    </NamespaceUris>
    <Aliases>
        <Alias Alias="Boolean">i=1</Alias>
        <Alias Alias="UInt32">i=7</Alias>
        <Alias Alias="String">i=12</Alias>
        <Alias Alias="HasModellingRule">i=37</Alias>
        <Alias Alias="HasTypeDefinition">i=40</Alias>
        <Alias Alias="HasSubtype">i=45</Alias>
        <Alias Alias="HasProperty">i=46</Alias>
        <Alias Alias="HasComponent">i=47</Alias>
        <Alias Alias="Argument">i=296</Alias>
    </Aliases>
    <Extensions>
        <Extension>
            <ModelInfo Tool="UaModeler" Hash="Zs8w1AQI71W8P/GOk3k/xQ=="
                       Version="1.3.4"/>
        </Extension>
    </Extensions>
```

```xml
<UAReferenceType NodeId="ns=1;i=4001" BrowseName="1:providesInputTo">
    <DisplayName>providesInputTo</DisplayName>
    <References>
        <Reference ReferenceType="HasSubtype" IsForward="false">
            i=33
        </Reference>
    </References>
    <InverseName Locale="en-US">inputProcidedBy</InverseName>
</UAReferenceType>
<UAObjectType IsAbstract="true" NodeId="ns=1;i=1001"
            BrowseName="1:FieldDevice">
    <DisplayName>FieldDevice</DisplayName>
    <References>
        <Reference ReferenceType="HasSubtype" IsForward="false">
            i=58
        </Reference>
        <Reference ReferenceType="HasComponent">ns=1;i=6001</Reference>
        <Reference ReferenceType="HasComponent">ns=1;i=6002</Reference>
    </References>
</UAObjectType>
<UAVariable DataType="String" ParentNodeId="ns=1;i=1001"
            NodeId="ns=1;i=6001" BrowseName="1:ManufacturerName"
            UserAccessLevel="3" AccessLevel="3">
    <DisplayName>ManufacturerName</DisplayName>
    <References>
        <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
        <Reference ReferenceType="HasModellingRule">i=78</Reference>
        <Reference ReferenceType="HasComponent" IsForward="false">
            ns=1;i=1001
        </Reference>
    </References>
</UAVariable>
<UAVariable DataType="String" ParentNodeId="ns=1;i=1001"
            NodeId="ns=1;i=6002" BrowseName="1:ModelName"
            UserAccessLevel="3" AccessLevel="3">
    <DisplayName>ModelName</DisplayName>
    <References>
        <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
        <Reference ReferenceType="HasModellingRule">i=78</Reference>
        <Reference ReferenceType="HasComponent" IsForward="false">
            ns=1;i=1001
        </Reference>
    </References>
</UAVariable>
<UAObjectType NodeId="ns=1;i=1002" BrowseName="1:Pump">
    <DisplayName>Pump</DisplayName>
    <References>
        <Reference ReferenceType="HasComponent">ns=1;i=6003</Reference>
        <Reference ReferenceType="HasComponent">ns=1;i=6004</Reference>
        <Reference ReferenceType="HasSubtype" IsForward="false">
            ns=1;i=1001
        </Reference>
        <Reference ReferenceType="HasComponent">ns=1;i=7001</Reference>
        <Reference ReferenceType="HasComponent">ns=1;i=7002</Reference>
    </References>
</UAObjectType>
<UAVariable DataType="Boolean" ParentNodeId="ns=1;i=1002"
            NodeId="ns=1;i=6003" BrowseName="1:isOn" UserAccessLevel="3"
            AccessLevel="3">
    <DisplayName>isOn</DisplayName>
    <References>
```

```xml
                <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
                <Reference ReferenceType="HasModellingRule">i=78</Reference>
                <Reference ReferenceType="HasComponent" IsForward="false">
                    ns=1;i=1002
                </Reference>
            </References>
        </UAVariable>
        <UAVariable DataType="UInt32" ParentNodeId="ns=1;i=1002"
                    NodeId="ns=1;i=6004" BrowseName="1:MotorRPM"
                    UserAccessLevel="3" AccessLevel="3">
            <DisplayName>MotorRPM</DisplayName>
            <References>
                <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
                <Reference ReferenceType="HasModellingRule">i=78</Reference>
                <Reference ReferenceType="HasComponent" IsForward="false">
                    ns=1;i=1002
                </Reference>
            </References>
        </UAVariable>
        <UAMethod ParentNodeId="ns=1;i=1002" NodeId="ns=1;i=7001"
                  BrowseName="1:startPump">
            <DisplayName>startPump</DisplayName>
            <References>
                <Reference ReferenceType="HasModellingRule">i=78</Reference>
                <Reference ReferenceType="HasProperty">ns=1;i=6005</Reference>
                <Reference ReferenceType="HasComponent" IsForward="false">
                    ns=1;i=1002
                </Reference>
            </References>
        </UAMethod>
        <UAVariable DataType="Argument" ParentNodeId="ns=1;i=7001" ValueRank="1"
                    NodeId="ns=1;i=6005" ArrayDimensions="1"
                    BrowseName="OutputArguments">
            <DisplayName>OutputArguments</DisplayName>
            <References>
                <Reference ReferenceType="HasModellingRule">i=78</Reference>
                <Reference ReferenceType="HasProperty"
                           IsForward="false">ns=1;i=7001</Reference>
                <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
            </References>
            <Value>
                <ListOfExtensionObject>
                    <ExtensionObject>
                        <TypeId>
                            <Identifier>i=297</Identifier>
                        </TypeId>
                        <Body>
                            <Argument>
                                <Name>started</Name>
                                <DataType>
                                    <Identifier>i=1</Identifier>
                                </DataType>
                                <ValueRank>-1</ValueRank>
                                <ArrayDimensions></ArrayDimensions>
                                <Description/>
                            </Argument>
                        </Body>
                    </ExtensionObject>
                </ListOfExtensionObject>
            </Value>
        </UAVariable>
```

```xml
    <UAMethod ParentNodeId="ns=1;i=1002" NodeId="ns=1;i=7002"
              BrowseName="1:stopPump">
        <DisplayName>stopPump</DisplayName>
        <References>
            <Reference ReferenceType="HasModellingRule">i=78</Reference>
            <Reference ReferenceType="HasProperty">ns=1;i=6006</Reference>
            <Reference ReferenceType="HasComponent"
                       IsForward="false">ns=1;i=1002</Reference>
        </References>
    </UAMethod>
    <UAVariable DataType="Argument" ParentNodeId="ns=1;i=7002" ValueRank="1"
                NodeId="ns=1;i=6006" ArrayDimensions="1"
                BrowseName="OutputArguments">
        <DisplayName>OutputArguments</DisplayName>
        <References>
            <Reference ReferenceType="HasModellingRule">i=78</Reference>
            <Reference ReferenceType="HasProperty" IsForward="false">
                ns=1;i=7002
            </Reference>
            <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
        </References>
        <Value>
            <ListOfExtensionObject>
                <ExtensionObject>
                    <TypeId>
                        <Identifier>i=297</Identifier>
                    </TypeId>
                    <Body>
                        <Argument>
                            <Name>stopped</Name>
                            <DataType>
                                <Identifier>i=1</Identifier>
                            </DataType>
                            <ValueRank>-1</ValueRank>
                            <ArrayDimensions></ArrayDimensions>
                            <Description/>
                        </Argument>
                    </Body>
                </ExtensionObject>
            </ListOfExtensionObject>
        </Value>
    </UAVariable>
</UANodeSet>
```

Take the previous snippet and save it to a file myNS.xml. To compile this nodeset into the corresponding C code, which can then be used by the open62541 stack, the nodeset compiler needs some arguments when you call it. The output of the help command gives you the following info:

```
$ python ./nodeset_compiler.py -h
usage: nodeset_compiler.py [-h] [-e <existingNodeSetXML>] [-x <nodeSetXML>]
                           [--generate-ns0] [--internal-headers]
                           [-b <blacklistFile>] [-i <ignoreFile>]
                           [-t <typesArray>]
                           [--max-string-length MAX_STRING_LENGTH] [-v]
                           <outputFile>


positional arguments:
  <outputFile>           The path/basename for the <output file>.c and <output
                         file>.h files to be generated. This will also be the
                         function name used in the header and c-file.
```

```
optional arguments:
  -h, --help            show this help message and exit
  -e <existingNodeSetXML>, --existing <existingNodeSetXML>
                        NodeSet XML files with nodes that are already present
                        on the server.
  -x <nodeSetXML>, --xml <nodeSetXML>
                        NodeSet XML files with nodes that shall be generated.
  --generate-ns0        Omit some consistency checks for bootstrapping
                        namespace 0, create references to parents and type
                        definitions manually
  --internal-headers    Include internal headers instead of amalgamated header
  -b <blacklistFile>, --blacklist <blacklistFile>
                        Loads a list of NodeIDs stored in blacklistFile (one
                        NodeID per line). Any of the nodeIds encountered in
                        this file will be removed from the nodeset prior to
                        compilation. Any references to these nodes will also
                        be removed
  -i <ignoreFile>, --ignore <ignoreFile>
                        Loads a list of NodeIDs stored in ignoreFile (one
                        NodeID per line). Any of the nodeIds encountered in
                        this file will be kept in the nodestore but not
                        printed in the generated code
  -t <typesArray>, --types-array <typesArray>
                        Types array for the given namespace. Can be used
                        mutliple times to define (in the same order as the
                        .xml files, first for --existing, then --xml) the type
                        arrays
  --max-string-length MAX_STRING_LENGTH
                        Maximum allowed length of a string literal. If longer,
                        it will be set to an empty string
  -v, --verbose         Make the script more verbose. Can be applied up to 4
                        times
```

So the resulting call looks like this:

```
$ python ./nodeset_compiler.py --types-array=UA_TYPES --existing ../../deps/ua-
→nodeset/Schema/Opc.Ua.NodeSet2.xml --xml myNS.xml myNS
```

And the output of the command:

```
INFO:__main__:Preprocessing (existing) ../../deps/ua-nodeset/Schema/Opc.Ua.
→NodeSet2.xml
INFO:__main__:Preprocessing myNS.xml
INFO:__main__:Generating Code
INFO:__main__:NodeSet generation code successfully printed
```

The first argument `--types-array=UA_TYPES` defines the name of the global array in open62541 which contains the corresponding types used within the nodeset in `NodeSet2.xml`. If you do not define your own datatypes, you can always use the `UA_TYPES` value. More on that later in this tutorial. The next argument `--existing ../../deps/ua-nodeset/Schema/Opc.Ua.NodeSet2.xml` points to the XML definition of the standard-defined namespace 0 (NS0). Namespace 0 is assumed to be loaded beforehand and provides definitions for data type, reference types, and so. Since we reference nodes from NS0 in our myNS.xml we need to tell the nodeset compiler that it should also load that nodeset, but not compile it into the output. Note that you may need to initialize the git submodule to get the `deps/ua-nodeset` folder (`git submodule --init --update`) or download the full `NodeSet2.xml` manually. The argument `--xml myNS.xml` points to the user-defined information model, whose nodes will be added to the abstract syntax tree. The script will then create the files `myNS.c` and `myNS.h` (indicated by the last argument `myNS`) containing the C code necessary to instantiate those namespaces.

Although it is possible to run the compiler this way, it is highly discouraged. If you care to examine the CMake-Lists.txt (examples/nodeset/CMakeLists.txt), you will find out that the file `server_nodeset.xml` is compiled

with the command:

```
COMMAND ${PYTHON_EXECUTABLE} ${PROJECT_SOURCE_DIR}/tools/nodeset_compiler/nodeset_
↪compiler.py
 --types-array=UA_TYPES
 --existing ${UA_FILE_NS0}
 --xml ${PROJECT_SOURCE_DIR}/examples/nodeset/server_nodeset.xml
 ${PROJECT_BINARY_DIR}/src_generated/example_nodeset
```

If you look into the files generated by the nodeset compiler, you will see that it generated a method called `extern UA_StatusCode myNS(UA_Server *server);`. You need to include the header and source file and then call the `myNS(server)` method right after creating the server instance with `UA_Server_new`. This will automatically add all the nodes to the server and return `UA_STATUSCODE_GOOD` if there weren't any errors. Additionally you need to compile the open62541 stack with the full NS0 by setting `UA_ENABLE_FULL_NS0=ON` in CMake. Otherwise the stack uses a minimal subset where many nodes are not included and thus adding a custom nodeset may fail.

This is how you can use the nodeset compiler to compile simple NodeSet XMLs to be used by the open62541 stack.

11.2 Creating object instances

One of the key benefits of defining object types is being able to create object instances fairly easily. Object instantiation is handled automatically when the typedefinition NodeId points to a valid ObjectType node. All Attributes and Methods contained in the objectType definition will be instantiated along with the object node.

While variables are copied from the objetType definition (allowing the user for example to attach new dataSources to them), methods are always only linked. This paradigm is identical to languages like C++: The method called is always the same piece of code, but the first argument is a pointer to an object. Likewise, in OPC UA, only one methodCallback can be attached to a specific methodNode. If that methodNode is called, the parent objectId will be passed to the method - it is the methods job to derefence which object instance it belongs to in that moment.

Let's look at an example that will create a pump instance given the newly defined objectType from myNS.xml:

```c
/* This work is licensed under a Creative Commons CCZero 1.0 Universal License.
 * See http://creativecommons.org/publicdomain/zero/1.0/ for more information. */

#include <signal.h>
#include <stdio.h>
#include "open62541.h"

/* Files myNS.h and myNS.c are created from myNS.xml */
#include "myNS.h"

UA_Boolean running = true;

static void stopHandler(int sign) {
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_SERVER, "received ctrl-c");
    running = false;
}

int main(int argc, char **argv) {
    signal(SIGINT, stopHandler);
    signal(SIGTERM, stopHandler);

    UA_ServerConfig *config = UA_ServerConfig_new_default();
    UA_Server *server = UA_Server_new(config);

    UA_StatusCode retval;
    /* create nodes from nodeset */
```

(continues on next page)

```c
    if (myNS(server) != UA_STATUSCODE_GOOD) {
        UA_LOG_ERROR(UA_Log_Stdout, UA_LOGCATEGORY_SERVER, "Could not add the␣
→example nodeset. "
            "Check previous output for any error.");
        retval = UA_STATUSCODE_BADUNEXPECTEDERROR;
    } else {


        UA_NodeId createdNodeId;
        UA_ObjectAttributes object_attr = UA_ObjectAttributes_default;

        object_attr.description = UA_LOCALIZEDTEXT("en-US", "A pump!");
        object_attr.displayName = UA_LOCALIZEDTEXT("en-US", "Pump1");

        // we assume that the myNS nodeset was added in namespace 2.
        // You should always use UA_Server_addNamespace to check what the
        // namespace index is for a given namespace URI. UA_Server_addNamespace
        // will just return the index if it is already added.
        UA_Server_addObjectNode(server, UA_NODEID_NUMERIC(1, 0),
                                UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER),
                                UA_NODEID_NUMERIC(0, UA_NS0ID_ORGANIZES),
                                UA_QUALIFIEDNAME(1, "Pump1"),
                                UA_NODEID_NUMERIC(2, 1002),
                                object_attr, NULL, &createdNodeId);


        retval = UA_Server_run(server, &running);
    }
    UA_Server_delete(server);
    UA_ServerConfig_delete(config);
    return (int) retval;
}
```

Make sure you have updated the headers and libs in your project, then recompile and run the server. Make especially sure you have added `myNS.h` to your include folder.

As you can see instantiating an object is not much different from creating an object node. The main difference is that you *must* use an objectType node as typeDefinition.

If you start the server and inspect the nodes with UA Expert, you will find the pump in the objects folder, which look like this Fig. 11.1.

As you can see the pump has inherited it's parents attributes (ManufacturerName and ModelName). Methods, in contrast to objects and variables, are never cloned but instead only linked. The reason is that you will quite propably attach a method callback to a central method, not each object. Objects are instantiated if they are *below* the object you are creating, so any object (like an object called associatedServer of ServerType) that is part of pump will be instantiated as well. Objects *above* you object are never instantiated, so the same ServerType object in Fielddevices would have been ommitted (the reason is that the recursive instantiation function protects itself from infinite recursions, which are hard to track when first ascending, then redescending into a tree).

11.3 Combination of multiple nodesets

In previous section you have seen how you can use the nodeset compiler with one single nodeset which depends on the default nodeset (NS0) `Opc.Ua.NodeSet2.xml`. The nodeset compiler also supports nodesets which depend on more than one nodeset. We will show this use-case with the PLCopen nodeset. The PLCopen nodeset `Opc.Ua.Plc.NodeSet2.xml` depends on the DI nodeset `Opc.Ua.Di.NodeSet2.xml` which then depends on NS0. This example is also shown in `examples/nodeset/CMakeLists.txt`.

For every nodeset we are depending, we need to call the nodeset compiler, bottom-up. The first nodeset is NS0. This is automatically created by the stack if you enable `UA_ENABLE_FULL_NS0=ON` in CMake. So we do not
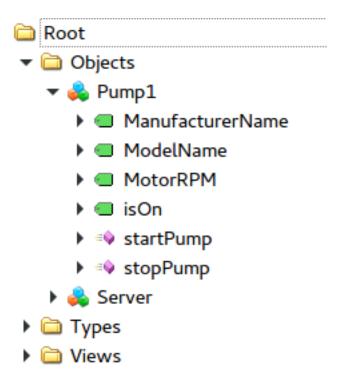
Fig. 11.1: Instantiated Pump Object with inherited children

need to care about that. If you look into the main CMakeLists.txt you can find the corresponding call to the nodeset compiler. It uses some additional flags which should only be used for NS0 and are not required for other nodesets.

The next nodeset in the list is the DI nodeset. This nodeset defines some additional data types in `deps/ua-nodeset/DI/Opc.Ua.Di.Types.bsd`. Since we also need these types within the generated code, we first need to compile the types into C code. The generated code is mainly a definition of the binary representation of the types required for encoding and decoding. The generation can be done using the `tools/generate_datatypes.py` script:

```
COMMAND ${PYTHON_EXECUTABLE} ${PROJECT_SOURCE_DIR}/tools/generate_datatypes.py
--namespace=2
--type-csv=${PROJECT_SOURCE_DIR}/deps/ua-nodeset/DI/OpcUaDiModel.csv
--type-bsd=${PROJECT_SOURCE_DIR}/deps/ua-nodeset/DI/Opc.Ua.Di.Types.bsd
--no-builtin
${PROJECT_BINARY_DIR}/src_generated/ua_types_di
```

The `namespace` parameter indicates the namespace index of the generated node IDs for the type definitions. Currently we need to rely that the namespace is also added at this position in the final server. There is no automatic inferring yet (pull requests are warmly welcome). The CSV and BSD files contain the metadata and definition for the types. The `--no-builtin` argument tells the script to skip internal datatypes which are always included in the stack. The last parameter is the output file and at the same time the name of the types array: `UA_TYPES_DI`.

Now you can compile the DI nodeset XML using the following command:

```
COMMAND ${PYTHON_EXECUTABLE} ${PROJECT_SOURCE_DIR}/tools/nodeset_compiler/nodeset_
↪compiler.py
--internal-headers
--types-array=UA_TYPES
--types-array=UA_TYPES_DI
--existing ${PROJECT_SOURCE_DIR}/deps/ua-nodeset/Schema/Opc.Ua.NodeSet2.xml
--xml ${PROJECT_SOURCE_DIR}/deps/ua-nodeset/DI/Opc.Ua.Di.NodeSet2.xml
${PROJECT_BINARY_DIR}/src_generated/ua_namespace_di
```

There are now two new arguments: `--internal-headers` indicates that internal headers (and non public API) should be included within the generated source code. This is currently required for nodesets which use structures as data values, and will probably be fixed in the future. Then you see two times the `--types-array` argument. The types array argument is matched with the nodesets in the same order as they appear on the command line: first the `existing` ones, and then the `xml`. It tells the nodeset compiler which types array it should use: `UA_TYPES` for `Opc.Ua.NodeSet2.xml` and `UA_TYPES_DI` for `Opc.Ua.Di.NodeSet2.xml`. This is the type array generated by the `generate_datatypes.py` script. The rest is similar to the example in previous section: `Opc.Ua.NodeSet2.xml` is assumed to exist already and only needs to be loaded for consistency checks, `Opc.Ua.Di.NodeSet2.xml` will be generated in the output file `ua_namespace_di.c/.h`

Next we can generate the PLCopen nodeset. Since it doesn't require any additional datatype definitions, we can immediately start with the nodeset compiler command:

```
COMMAND ${PYTHON_EXECUTABLE} ${PROJECT_SOURCE_DIR}/tools/nodeset_compiler/nodeset_
→compiler.py
--internal-headers
--types-array=UA_TYPES
--types-array=UA_TYPES_DI
# PLCopen has no specific type definition, thus use the default UA_TYPES to ignore␣
→it
--types-array=UA_TYPES
--existing ${PROJECT_SOURCE_DIR}/deps/ua-nodeset/Schema/Opc.Ua.NodeSet2.xml
--existing ${PROJECT_SOURCE_DIR}/deps/ua-nodeset/DI/Opc.Ua.Di.NodeSet2.xml
--xml ${PROJECT_SOURCE_DIR}/deps/ua-nodeset/PLCopen/Opc.Ua.Plc.NodeSet2.xml
${PROJECT_BINARY_DIR}/src_generated/ua_namespace_plc
```

This call is quite similar to the compilation of the DI nodeset. As you can see, we do not define any specific types array for the PLCopen nodeset, but just use `UA_TYPES` to ignore it. Since the PLCopen nodeset depends on the NS0 and DI nodeset, we need to tell the nodeset compiler that these two nodesets should be seen as already existing. Make sure that the order is the same as in your XML file, e.g., in this case the order indicated in `Opc.Ua.Plc.NodeSet2.xml -> UANodeSet -> Models -> Model`.

As a result of the previous scripts you will have multiple source files:

- ua_types_di_generated.c
- ua_types_di_generated.h
- ua_types_di_generated_encoding_binary.h
- ua_types_di_generated_handling.h
- ua_namespace_di.c
- ua_namespace_di.h
- ua_namespace_plc.c
- ua_namespace_plc.h

Finally you need to include all these files in your build process and call the corresponding initialization methods for the nodesets. An example application could look like this:

```
UA_ServerConfig *config = UA_ServerConfig_new_default();
UA_Server *server = UA_Server_new(config);

/* create nodes from nodeset */
UA_StatusCode retval = ua_namespace_di(server);
if(retval != UA_STATUSCODE_GOOD) {
    UA_LOG_ERROR(UA_Log_Stdout, UA_LOGCATEGORY_SERVER, "Adding the DI namespace␣
→failed. Please check previous error output.");
    UA_Server_delete(server);
    UA_ServerConfig_delete(config);
    return (int)UA_STATUSCODE_BADUNEXPECTEDERROR;
}
```

(continues on next page)

```c
retval |= ua_namespace_plc(server);
if(retval != UA_STATUSCODE_GOOD) {
    UA_LOG_ERROR(UA_Log_Stdout, UA_LOGCATEGORY_SERVER, "Adding the PLCopen
↪namespace failed. Please check previous error output.");
    UA_Server_delete(server);
    UA_ServerConfig_delete(config);
    return (int)UA_STATUSCODE_BADUNEXPECTEDERROR;
}

retval = UA_Server_run(server, &running);
```

Internals

12.1 Networking Plugin API

12.1.1 Connection

Client-server connections are represented by a *UA_Connection*. The connection is stateful and stores partially received messages, and so on. In addition, the connection contains function pointers to the underlying networking implementation. An example for this is the *send* function. So the connection encapsulates all the required networking functionality. This lets users on embedded (or otherwise exotic) systems implement their own networking plugins with a clear interface to the main open62541 library.

```c
typedef struct {
    UA_UInt32 protocolVersion;
    UA_UInt32 sendBufferSize;
    UA_UInt32 recvBufferSize;
    UA_UInt32 maxMessageSize;
    UA_UInt32 maxChunkCount;
} UA_ConnectionConfig;

typedef enum {
    UA_CONNECTION_CLOSED,        /* The socket has been closed and the connection
                                  * will be deleted */
    UA_CONNECTION_OPENING,       /* The socket is open, but the HEL/ACK handshake
                                  * is not done */
    UA_CONNECTION_ESTABLISHED    /* The socket is open and the connection
                                  * configured */
} UA_ConnectionState;

struct UA_Connection {
    UA_ConnectionState state;
    UA_ConnectionConfig localConf;
    UA_ConnectionConfig remoteConf;
    UA_SecureChannel *channel;       /* The securechannel that is attached to
                                      * this connection */
    UA_Int32 sockfd;                 /* Most connectivity solutions run on
                                      * sockets. Having the socket id here
                                      * simplifies the design. */
    UA_DateTime openingDate;         /* The date the connection was created */
```

```
    void *handle;                    /* A pointer to internal data */
    UA_ByteString incompleteMessage; /* A half-received message (TCP is a
                                      * streaming protocol) is stored here */

    /* Get a buffer for sending */
    UA_StatusCode (*getSendBuffer)(UA_Connection *connection, size_t length,
                                   UA_ByteString *buf);

    /* Release the send buffer manually */
    void (*releaseSendBuffer)(UA_Connection *connection, UA_ByteString *buf);

    /* Sends a message over the connection. The message buffer is always freed,
     * even if sending fails.
     *
     * @param connection The connection
     * @param buf The message buffer
     * @return Returns an error code or UA_STATUSCODE_GOOD. */
    UA_StatusCode (*send)(UA_Connection *connection, UA_ByteString *buf);

    /* Receive a message from the remote connection
     *
     * @param connection The connection
     * @param response The response string. It is allocated by the connection
     *        and needs to be freed with connection->releaseBuffer
     * @param timeout Timeout of the recv operation in milliseconds
     * @return Returns UA_STATUSCODE_BADCOMMUNICATIONERROR if the recv operation
     *         can be repeated, UA_STATUSCODE_GOOD if it succeeded and
     *         UA_STATUSCODE_BADCONNECTIONCLOSED if the connection was
     *         closed. */
    UA_StatusCode (*recv)(UA_Connection *connection, UA_ByteString *response,
                          UA_UInt32 timeout);

    /* Release the buffer of a received message */
    void (*releaseRecvBuffer)(UA_Connection *connection, UA_ByteString *buf);

    /* Close the connection. The network layer closes the socket. This is picked
     * up during the next 'listen' and the connection is freed in the network
     * layer. */
    void (*close)(UA_Connection *connection);

    /* To be called only from within the server (and not the network layer).
     * Frees up the connection's memory. */
    void (*free)(UA_Connection *connection);
};

/* Cleans up half-received messages, and so on. Called from connection->free. */
void
UA_Connection_deleteMembers(UA_Connection *connection);
```

12.1.2 Server Network Layer

The server exposes two functions to interact with remote clients: *processBinaryMessage* and *removeConnection*. These functions are called by the server network layer.

It is the job of the server network layer to listen on a TCP socket, to accept new connections, to call the server with received messages and to signal closed connections to the server.

The network layer is part of the server config. So users can provide a custom implementation if the provided example does not fit their architecture. The network layer is invoked only from the server's main loop. So the network layer does not need to be thread-safe. If the networklayer receives a positive duration for blocking

listening, the server's main loop will block until a message is received or the duration times out.

```c
/* Process a binary message (TCP packet). The message can contain partial
 * chunks. (TCP is a streaming protocol and packets may be split/merge during
 * transport.) After processing, the message is freed with
 * connection->releaseRecvBuffer. */
void
UA_Server_processBinaryMessage(UA_Server *server, UA_Connection *connection,
                               UA_ByteString *message);

/* The server internally cleans up the connection and then calls
 * connection->free. */
void
UA_Server_removeConnection(UA_Server *server, UA_Connection *connection);

struct UA_ServerNetworkLayer {
    void *handle; /* Internal data */
    UA_String discoveryUrl;

    /* Start listening on the networklayer.
     *
     * @param nl The network layer
     * @return Returns UA_STATUSCODE_GOOD or an error code. */
    UA_StatusCode (*start)(UA_ServerNetworkLayer *nl, const UA_String
→*customHostname);

    /* Listen for new and closed connections and arriving packets. Calls
     * UA_Server_processBinaryMessage for the arriving packets. Closed
     * connections are picked up here and forwarded to
     * UA_Server_removeConnection where they are cleaned up and freed.
     *
     * @param nl The network layer
     * @param server The server for processing the incoming packets and for
     *               closing connections.
     * @param timeout The timeout during which an event must arrive in
     *                milliseconds
     * @return A statuscode for the status of the network layer. */
    UA_StatusCode (*listen)(UA_ServerNetworkLayer *nl, UA_Server *server,
                            UA_UInt16 timeout);

    /* Close the network socket and all open connections. Afterwards, the
     * network layer can be safely deleted.
     *
     * @param nl The network layer
     * @param server The server that processes the incoming packets and for
     *               closing connections before deleting them.
     * @return A statuscode for the status of the closing operation. */
    void (*stop)(UA_ServerNetworkLayer *nl, UA_Server *server);

    /* Deletes the network layer context. Call only after stopping. */
    void (*deleteMembers)(UA_ServerNetworkLayer *nl);
};
```

12.1.3 Client Network Layer

The client has only a single connection used for sending and receiving binary messages.

```c
/* @param localConf the connection config for this client
 * @param endpointUrl to where to connect
 * @param timeout in ms until the connection try times out if remote not reachable
 * @param logger the logger to use */
```

(continues on next page)

```
typedef UA_Connection
(*UA_ConnectClientConnection)(UA_ConnectionConfig localConf, const char␣
↪*endpointUrl,
                              const UA_UInt32 timeout, UA_Logger logger);
```

12.1.4 Endpoint URL Parser

The endpoint URL parser is generally useful for the implementation of network layer plugins.

```
/* Split the given endpoint url into hostname, port and path. All arguments must
 * be non-NULL. EndpointUrls have the form "opc.tcp://hostname:port/path", port
 * and path may be omitted (together with the prefix colon and slash).
 *
 * @param endpointUrl The endpoint URL.
 * @param outHostname Set to the parsed hostname. The string points into the
 *        original endpointUrl, so no memory is allocated. If an IPv6 address is
 *        given, hostname contains e.g. '[2001:0db8:85a3::8a2e:0370:7334]'
 * @param outPort Set to the port of the url or left unchanged.
 * @param outPath Set to the path if one is present in the endpointUrl.
 *        Starting or trailing '/' are NOT included in the path. The string
 *        points into the original endpointUrl, so no memory is allocated.
 * @return Returns UA_STATUSCODE_BADTCPENDPOINTURLINVALID if parsing failed. */
UA_StatusCode
UA_parseEndpointUrl(const UA_String *endpointUrl, UA_String *outHostname,
                    UA_UInt16 *outPort, UA_String *outPath);
```

12.2 Access Control Plugin API

The access control callback is used to authenticate sessions and grant access rights accordingly.

```
struct UA_AccessControl;
typedef struct UA_AccessControl UA_AccessControl;

struct UA_AccessControl {
    void *context;
    void (*deleteMembers)(UA_AccessControl *ac);

    /* Supported login mechanisms. The server endpoints are created from here. */
    size_t userTokenPoliciesSize;
    UA_UserTokenPolicy *userTokenPolicies;

    /* Authenticate a session. The session context is attached to the session and
     * later passed into the node-based access control callbacks. */
    UA_StatusCode (*activateSession)(UA_Server *server, UA_AccessControl *ac,
                                     const UA_NodeId *sessionId,
                                     const UA_ExtensionObject *userIdentityToken,
                                     void **sessionContext);

    /* Deauthenticate a session and cleanup */
    void (*closeSession)(UA_Server *server, UA_AccessControl *ac,
                         const UA_NodeId *sessionId, void *sessionContext);

    /* Access control for all nodes*/
    UA_UInt32 (*getUserRightsMask)(UA_Server *server, UA_AccessControl *ac,
                                   const UA_NodeId *sessionId, void␣
↪*sessionContext,
                                   const UA_NodeId *nodeId, void *nodeContext);
```

```c
    /* Additional access control for variable nodes */
    UA_Byte (*getUserAccessLevel)(UA_Server *server, UA_AccessControl *ac,
                                  const UA_NodeId *sessionId, void *sessionContext,
                                  const UA_NodeId *nodeId, void *nodeContext);

    /* Additional access control for method nodes */
    UA_Boolean (*getUserExecutable)(UA_Server *server, UA_AccessControl *ac,
                                    const UA_NodeId *sessionId, void
↪*sessionContext,
                                    const UA_NodeId *methodId, void
↪*methodContext);

    /* Additional access control for calling a method node in the context of a
     * specific object */
    UA_Boolean (*getUserExecutableOnObject)(UA_Server *server, UA_AccessControl
↪*ac,
                                            const UA_NodeId *sessionId, void
↪*sessionContext,
                                            const UA_NodeId *methodId, void
↪*methodContext,
                                            const UA_NodeId *objectId, void
↪*objectContext);

    /* Allow adding a node */
    UA_Boolean (*allowAddNode)(UA_Server *server, UA_AccessControl *ac,
                               const UA_NodeId *sessionId, void *sessionContext,
                               const UA_AddNodesItem *item);

    /* Allow adding a reference */
    UA_Boolean (*allowAddReference)(UA_Server *server, UA_AccessControl *ac,
                                    const UA_NodeId *sessionId, void
↪*sessionContext,
                                    const UA_AddReferencesItem *item);

    /* Allow deleting a node */
    UA_Boolean (*allowDeleteNode)(UA_Server *server, UA_AccessControl *ac,
                                  const UA_NodeId *sessionId, void *sessionContext,
                                  const UA_DeleteNodesItem *item);

    /* Allow deleting a reference */
    UA_Boolean (*allowDeleteReference)(UA_Server *server, UA_AccessControl *ac,
                                       const UA_NodeId *sessionId, void
↪*sessionContext,
                                       const UA_DeleteReferencesItem *item);
};
```

12.3 Logging Plugin API

Servers and clients must define a logger in their configuration. The logger is just a function pointer. Every log-message consists of a log-level, a log-category and a string message content. The timestamp of the log-message is created within the logger.

```c
typedef enum {
    UA_LOGLEVEL_TRACE,
    UA_LOGLEVEL_DEBUG,
    UA_LOGLEVEL_INFO,
    UA_LOGLEVEL_WARNING,
```

```
    UA_LOGLEVEL_ERROR,
    UA_LOGLEVEL_FATAL
} UA_LogLevel;

typedef enum {
    UA_LOGCATEGORY_NETWORK,
    UA_LOGCATEGORY_SECURECHANNEL,
    UA_LOGCATEGORY_SESSION,
    UA_LOGCATEGORY_SERVER,
    UA_LOGCATEGORY_CLIENT,
    UA_LOGCATEGORY_USERLAND,
    UA_LOGCATEGORY_SECURITYPOLICY
} UA_LogCategory;
```

The message string and following varargs are formatted according to the rules of the printf command. Do not call the logger directly. Instead, make use of the convenience macros that take the minimum log-level defined in ua_config.h into account.

```
typedef void (*UA_Logger)(UA_LogLevel level, UA_LogCategory category,
                          const char *msg, va_list args);

static UA_INLINE UA_FORMAT(3,4) void
UA_LOG_TRACE(UA_Logger logger, UA_LogCategory category, const char *msg, ...) {
#if UA_LOGLEVEL <= 100
    va_list args; va_start(args, msg);
    logger(UA_LOGLEVEL_TRACE, category, msg, args);
    va_end(args);
#endif
}

static UA_INLINE UA_FORMAT(3,4) void
UA_LOG_DEBUG(UA_Logger logger, UA_LogCategory category, const char *msg, ...) {
#if UA_LOGLEVEL <= 200
    va_list args; va_start(args, msg);
    logger(UA_LOGLEVEL_DEBUG, category, msg, args);
    va_end(args);
#endif
}

static UA_INLINE UA_FORMAT(3,4) void
UA_LOG_INFO(UA_Logger logger, UA_LogCategory category, const char *msg, ...) {
#if UA_LOGLEVEL <= 300
    va_list args; va_start(args, msg);
    logger(UA_LOGLEVEL_INFO, category, msg, args);
    va_end(args);
#endif
}

static UA_INLINE UA_FORMAT(3,4) void
UA_LOG_WARNING(UA_Logger logger, UA_LogCategory category, const char *msg, ...) {
#if UA_LOGLEVEL <= 400
    va_list args; va_start(args, msg);
    logger(UA_LOGLEVEL_WARNING, category, msg, args);
    va_end(args);
#endif
}

static UA_INLINE UA_FORMAT(3,4) void
UA_LOG_ERROR(UA_Logger logger, UA_LogCategory category, const char *msg, ...) {
#if UA_LOGLEVEL <= 500
    va_list args; va_start(args, msg);
```

```
    logger(UA_LOGLEVEL_ERROR, category, msg, args);
    va_end(args);
#endif
}

static UA_INLINE UA_FORMAT(3,4) void
UA_LOG_FATAL(UA_Logger logger, UA_LogCategory category, const char *msg, ...) {
#if UA_LOGLEVEL <= 600
    va_list args; va_start(args, msg);
    logger(UA_LOGLEVEL_FATAL, category, msg, args);
    va_end(args);
#endif
}
```

12.3.1 Convenience macros for complex types

```
#define UA_PRINTF_GUID_FORMAT "%08x-%04x-%04x-%02x%02x-%02x%02x%02x%02x%02x%02x"
#define UA_PRINTF_GUID_DATA(GUID) (GUID).data1, (GUID).data2, (GUID).data3, \
        (GUID).data4[0], (GUID).data4[1], (GUID).data4[2], (GUID).data4[3], \
        (GUID).data4[4], (GUID).data4[5], (GUID).data4[6], (GUID).data4[7]

#define UA_PRINTF_STRING_FORMAT "\"%.*s\""
#define UA_PRINTF_STRING_DATA(STRING) (int)(STRING).length, (STRING).data
```