SDK comparison

This page contains comparison between the three SDKs tested during the PoC. The SDKs are compared from different points of view. The sources of information are our own observations, online material and information sent to us.

Supported OPC UA features

https://github.com/open62541/open62541/blob/master/FEATURES.md

	open62541	UA HP	Matrikon FLEX	Softing
Server	Micro Embedded	Standard	Standard	Embedded
Client	x	v.1.3.0 (2019)	x	2020
Encryption - Server	x	x	x	x
Encryption - Client	v.0.4 (unknown date)	v.1.3.0 (march 2019) ?	x	?
Authentication	Anonymous User name - password	Anonymous User name - password X509	?	Anonymous User name - password
Data Access	x	x	x	x
Methods	x	x	x	x
Events	v.0.4 (unknown date)	x	x	Q3 2019
Alarms & Conditions	Pull request exists	unclear, not in v.1.3.0	x	Q3 2019
Historical Data	v.0.4 (unknown date)	-	x	2020
Pub/Sub	v.0.4 (unknown date)	2019	2019	x

Of the tested SDKs Matrikon FLEX has the most complete set of OPC UA features. Both open62541 and UA HP are rather new SDKs and many features are still in the pipeline. For more complete descriptions of what features each SDK offers, refer to the following web locations:

open62541:

https://open62541.org/doc/current/

Unified Automation High Performance OPC UA server client bundle:

https://www.unified-automation.com/products/server-sdk/highperf-ua-server-sdk.html

Matrikon FLEX OPC UA SDK:

https://www.matrikonopc.com/downloads/1129/index.aspx

Maturity and support

open62541:

+ open62541 is an open source project backed by an active community.
+ The project has several commercial and institutional partners.
+ The project has a good CI system and good documentation.
+/- SDK still under development, some of the non-released features tested during the PoC were somewhat unstable.

UA:

+ Unified Automation has good technical support, for example we received a small fix to the SDK within a week after discussing its need with them.
+ The SDK has decent documentation
+/- SDK still under development, client in beta-state and felt was a bit unstable
- Unclear to which extent the big-endian support is tested. We found and tested several issues related to big-endian support, all though related to the client functionality.

Unified Automation offers support only as part of paid support packages which contains a certain amount of incidents. Updates are delivered through yearly maintenance packages.

Matrikon:

Feature-wise the Matrikon SDK is the most complete of the ones compared and the server worked almost immediately with most PLCs.

Documentation is pretty poor. It describes the stack usage in very general way and the only way to create actual implementation is to view the examples. Code reference is in Doxygen format and is pretty sparse and often does not describe e.g. input arguments in any more detail than could be deduced from the argument name.

The examples are pretty good and usually give an idea where to look if you want to do something different than in the examples.

The core parts of the SDK is provided as obfuscated C code which sometimes made solving issues was difficult.

License terms

open62541:

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.

UA:

The UA HP SDK has a single seat source code developer license. This means that only named license holders can do active development using the SDK, i.e. program against its interfaces. The licenses are possible to transfer, either temporarily or permanently to other developers within Wärtsilä or to a subcontractor. The SDK has a one year warranty period.

Matrikon:

<will be studied and added upon need>

Resource usage

Naturally, all of the SDKs use some CPU - with more data transferred more CPU resources are needed. For details, please refer to the page with CPU load measurement. Generally speaking the difference in CPU load caused by the SDKs is not large, open62541 seems to use the least amount and Matrikon the biggest. Subscribing to data as a client seems to be more expensive than providing data as a server. Encrypting the communication causes a few extra percentage points of CPU load.

open62541:

Out of box, open62541 uses standard malloc/free and dynamically allocates memory during runtime but it is possible to replace these with own implementation. During the PoC we used the standard ones and we didn't have any problems with memory consumption, but this does introduce a risk as a memory leak in the SDK could bring the whole module down.

open62541 has an experimental multithreading option which distributes the work to a number of worker threads. We did not try that, but used one thread for the server and one thread for each client connection.

UA:

The UA HP SDK allocates memory from a memory pool defined at compile time. This means that some worst case usage scenario needs to be defined already when compiling the software. But makes the SDK safer to use than for example the open62541 SDK.

The design of the UA HP SDK is based on a single threaded design using asynchronous functions and callbacks. In the PoC we used the possibility to produce data for the server from different threads, in the same manner it should be possible connect the client to our process using separate threads. But the core functionality remains in one single thread and that makes things like for example prioritizing one client over another hard.

Matrikon:

Resource consumption of the stack seems to be very high. If stack is allowed to handle memory allocations by itself (UASDK_USE_SYSTEM_HEAP = 1) the hardware module will run out of memory as soon as the stack allocates itself. This can be remedied by using a custom memory pool (UAServer_t::CreateAllocator) but even then at least 24 MB of memory was needed to prevent OOM errors especially when multiple clients were connecting. 32MB was used during the testing.

Greater concern is the usage of locks. When implementing the UNIC interface for the library (for threading and mutexes) the system immediately ran out of mutexes. The used test configuration of about 1000 items allocated ~12000 lock objects during initialization while the maximum configured mutex amount in platform was 100. It was possible to run the configuration in virtual module when the platform limit was increased but hardware module was not able to allocate enough mutexes even then. It is unclear why the stack would allocate such an excessive amount of mutexes and is it possible to do anything about it.

The stack also seems to require lots of dynamic memory allocations, e.g. reading one DC code requires three allocations (data value object + actual value + timestamp).

Resource usage could possible be minimized by using the older C API implementation, but that does not support multi threading.

Diagnostics

All of the tested SDKs are able to produce diagnostic log messages to the console. The log levels are compile-time configurable. The open62541 SDK produces more understandable log messages than the UA HP stack. The UA HP stack is able to produce a memory usage report which is handy as the memory pool for this SDK needs to be configured by hand.

Conclusions

Of the tested SDKs the open62541 seems to have the best performance. In most cases it is very responsive and causes the least CPU load of the tested SDKs. However, it was not entirely stable and showed inconsistent behavior in some cases. This might of cause be related to the fact that we did not use any released version, but the master branch and also we did not have time to go to dept with all of the random problems encountered. One issue with the open62541 stack is its use of memory - that will need some work if the SDK is to be taken into use released software. Else, the open62541 SDK was the easiest to take into use and customize, partly because of the very good documentation provided.

The Matrikon Flex was very hard to take into use and it proved not to be possible to really optimize for our use - at least within the given timeframe and with the give resources. Possibly it is simply too large to properly fit into the COM-10 that was the target in our PoC. On the other hand, once running the Matrikon based server seemed very stable and it worked almost immediately with most PLCs. We did not have time to test a client based upon the Matrikon SDK.

The server part of the UA HP SDK is the most stable of the tested ones. The client is not nearly on the same level when it comes to stability but, as noted, the client is still in beta stage. From CPU load point-of-view the UA HP SDK is almost at par with the open62541 SDK. The biggest draw-back with the UA HP is the lack of support for multi-threading. This makes it very hard or possibly impossible to prioritize connections over each other. Unified Automation offers also an older C SDK an and a C++ SDK, both more mature and with multi-thread support. One possible option is to test one of them.