Contents

8 Using the OPC server

8.2 Optimum OPC mode for an application

The OPC interface for data access offers various possibilities for read/write access to the process variables regardless of whether the Custom or Automation Interface is in use. The access method selected influences the performance of the Simatic Net OPC server and the OPC client.

In the case of the variable services for S7 Communication and S5-compatible Communication via Industrial Ethernet (Send/Receive), access to variables can be accelerated through suitable structuring of the variables in the address area of the partner device.
The service selected can also affect the throughput achieved. For example, large data packets can be transferred with the block services of S7 Communication.

Please note the following tips in order to achieve the highest possible throughput with your application.

TIP 1: Use quantity operations
With many methods, the OPC interfaces facilitate transfer of large numbers of process variables in one field as parameters with one function call. The use of quantity operations is always beneficial for the throughput since fewer function calls and process changes are required between the OPC client and the OPC server, and the OPC server itself can optimize communication over the network, for example by combining individual jobs. The OPC server too uses quantity operations in the DataChange-Callback function for transferring change messages.

Examples of quantity calls with OPC Data Access:

IOPCItemMgt::AddItems	Combining many OPC items to a group
	TIP: Calling AddItems is faster if the group is NOT active. Don't activate the group until all OPC items have been added.
IOPCSyncIO::Read	Synchronous reading of value, time stamp and quality of many OPC items.
IOPCSyncIO::Write	Synchronous writing of value, time stamp and quality of many OPC items.

TIP 2: Accessing the OPC cache
The OPC server has a cache for all OPC items, that is, an internal intermediate memory holding the most recently acquired values of the OPC items. To allow the cache to be updated, it must be possible to successfully read the OPC item. For those OPC items that are monitored by the OPC server in an active group, the cache contents are updated in accordance with the OPC update rates.

With all protocols, access to the cache is significantly faster than access via the network.
Therefore, if you are monitoring an OPC item, and the values are sufficiently up-to-date due to the update rate, you should use accesses to the cache.

You must note, however, that the data in the cache also have to be updated and because of this, use of active items put a load on the computer processor and the communication system.

Examples of services that can be used on the cache:

IOPCSyncIO::Read	Synchronous reading of value, time stamp and quality of many OPC items from the cache.
IOPCAsyncIO::Refresh	A callback is generated for all active OPC items, regardless of value, and the value stored in the cache is returned.

TIP 3: Structuring of the items in the case of S7/SR variable access

The OPC server receives an optimization algorithm from SimaticNet for the variable services of S7 Communication and S5-compatible Communication via Ethernet (Send/Receive): several jobs for access to, say, individual bits or bytes are converted internally to a field access to the partner device. This reduces the number of data packets to be transported over the network, improves the capacity utilization of the individual packets, and increases the proportion of user data in a packet.

This optimization is applicable to both read and write accesses and is activated by default. This optimization algorithm is invisible to the OPC client.
So that the optimization can take effect, a few rules have to be observed concerning the arrangement of the OPC items in the address area.

- OPC items that are read or monitored simultaneously **should** be arranged successively in the address area of the partner device if possible. Smaller gaps between the relevant sections are also processed but they adversely affect data throughput.

Example: Organization of a data block for read access

DB10,W10	Executed by the communication system as one read field access to DB10,B10,12. (Please note that this is converted to a read job despite the gaps at bytes 18 and 19. The “superfluously” read data are rejected.)
DB10,B12	
DB10,B13	
DB10,DW14	
DB10,W20	

- OPC items that are written simultaneously **must** be arranged successively in the address area in order for optimization to take effect. Gaps cannot be accepted since the converted field is transferred complete to the partner device and therefore the address area of the gaps would be overwritten with undefined values. The OPC server then transmits individual job requests by means of the communication system.

Example: Organization of a data block for write access

DB10,W10	Is executed via the communication system as two write accesses to fields DB10,B10,8 and the individual variables DB10,W20.
DB10,B12	
DB10,B13	
DB10,DW14	
DB10,W20	

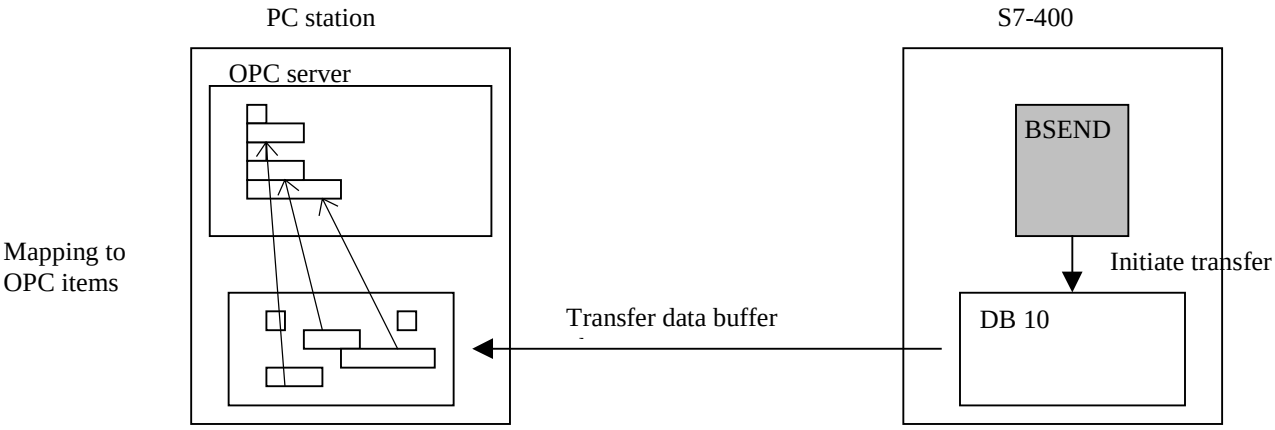
> **TIP**: If your OPC client does not rely on the use of individual OPC items for the process variables, you have the alternative of using fields for accessing the relevant data and of assigning the elements of the read fields to individual process variables.

TIP 4: Using the block services with S7 and SR

If large data packets are to be transferred to the partner device, S7 Communication and S5-compatible Communication offer a solution with the block services via Industrial Ethernet (SR). Using the block service, one partner sends a data packet to the other communications partners. Transfer of the data only loads the network when a partner explicitly transmits a send request. The OPC server of Simatic Net facilitates structuring of the data blocks: OPC items can be assigned to individual parts of the data packet.

Note: The S7 block services are only available to devices of the S7-400/M7 range, but the SR block services, in contrast, are available to almost all devices of the S5 and S7 ranges.

Application example: An S7-400 device sends a data packet to a PC station with S7 OPC server.



On the PC station side, several OPC items for receiving data buffers are inserted in an active group as active items. It is necessary to provide these receive items so that the receive buffers can be set up on the protocol side.

A user program initiates the function block BSEND on the S7-400 device. This triggers transfer of a data area to the PC station as a buffer.

The received buffer is transferred to the OPC server on the PC station. The OPC server then maps the buffer sub-areas defined by the ItemID of the OPC item to the relevant OPC item and sends a DataChange-Callback to the OPC client, if the client has so requested.

TIP 5: Use suitable OPC methods in your OPC clients
The OPC interface for data access offers the following data access methods:
- Synchronous read/write
- Asynchronous read/write
- Monitoring of variables

When designing and programming an OPC client, please note the characteristics of the operations as described below.

8.2.1 Synchronous read/write

A synchronous function call for accessing process data – whether read or write – only returns to the caller when communication over the network has been fully completed and the requested information has been received. When program control has returned to the caller, the desired information is transferred.

Advantages
- Simple programming.
- Overall higher data throughput, since there is only 1 process change per request between the OPC client and the OPC server.

Disadvantages:
- Execution of your own application is suspended while the synchronous request is processed. The request only returns when all data have been read. If the synchronous request is not transmitted in its own thread, this can cause interactive applications to block during execution of the request.

Parameters:
In the case of synchronous requests, data throughput does not depend on the parameters “Update rate” and “Cycle time”.

Summary:
Synchronous access should always be used in those cases where extended interruption of the user program is less significant, such as in the use of “worker threads”. Synchronous access always offers the fastest possible access to the data of the partner device.

8.2.2 Asynchronous read/write

An asynchronous function call for accessing process data returns to the caller immediately after transmission of the request to the OPC server. Return of program control only tells the OPC client whether or not the request has been successfully sent to the OPC server. The OPC server has assigned a “TransactionID” to the request that allows the client to later identify the answer to this request.

At a later undefined time, the OPC server calls the Callback function “AsyncReadComplete” or “AsyncWriteComplete” of the OPC client. The results of the previous read request and the TransactionID are now transferred to the client as the call parameters of the Callback. The time at which the request is executed, that is, the time at which the data are transferred via the network, is independent of the program sequence of the OPC client. Such behavior is called asynchronous.

Advantages:
- No extended interruptions of own applications since the request is processed in parallel with the application.

Disadvantages:
- The application program is somewhat more complicated than an implementation for synchronous calls since it must provide a callback mechanism that can receive the result of request processing at any time. However, Windows programs have asynchronous mechanisms as standard in order to be able to respond to user inputs.
- In particular, when transferring just a few variables in one request, a high overhead is created by process transitions in calling and callback. (Double the number of transitions as in synchronous calls)

Parameters:
In the case of asynchronous requests, data throughput does not depend on the parameters “Update rate” and “Cycle time”.

Summary:
Asynchronous accesses are only meaningful when large data volumes have to be read and the user program is to remain responsive during request processing. Asynchronous accesses to the cache of the OPC server are not meaningful. This results in high processor load simply through process transitions between OPC client and OPC server.

8.2.3 Monitoring of variables

Monitoring of variables is a central functionality of the OPC Data Access interface. When monitoring variables, the OPC server continuously checks to see if the value or the quality of the variables have changed.

To use the monitoring function via the OPC server, the OPC client adds active OPC items to a group and activates the group. All active OPC items in all active groups are monitored. The OPC client must also provide the “DataChange” Callback function.

If a change is detected by the OPC server, it calls the “DataChange” Callback function of the OPC client and transfers the changed OPC items, the values, the qualities and the time stamp as function parameters.

The OPC client is not loaded by monitoring of the variables. Only when a change has been detected is a process change made to the OPC client and the program code of the client executed. So that the OPC client is not overloaded with change messages in the case of rapidly changing process variables, it can use the group-specific parameter “Update Rate” to specify the minimum frequency with which it wants to be called.

> **TIP:** “Noisy” analog values would result in frequent change messages because the value is subject to continuous slight change. The noise can be filtered out using the group-specific parameter “PercentDeadBand”. Then only those changes are signaled that are greater than a percentage part of the value range of the variables. (Requirement: the value range of the variables has been defined in the Configurator symbol file.)

Advantages
- Application is only notified when the process data have changed.
- Overall higher data throughput since there is little process change and good optimization is possible depending on the makeup of the item structure.
- The variables to be monitored can be switched on and off by the client on a group basis.

Disadvantages:
- The response time between the change of a value in the process and transfer of the new value to the client is greater than the update rate of the group in each case.
- The user program is a little more complicated than a synchronous implementation since it requires an asynchronous section for receiving value changes. However, Windows programs have asynchronous mechanisms as standard in order to be able to respond to user inputs.

Parameters:
Data throughput is determined by the “Update Rate” and “Cycle Time” parameters of a group. The cycle time determines the smallest possible update rate. The update rate should be specified in integer multiples of the cycle time.

Summary:
Monitoring of variables is the optimum solution if an application requires constantly up-to-date data from the process or a section of the process.

