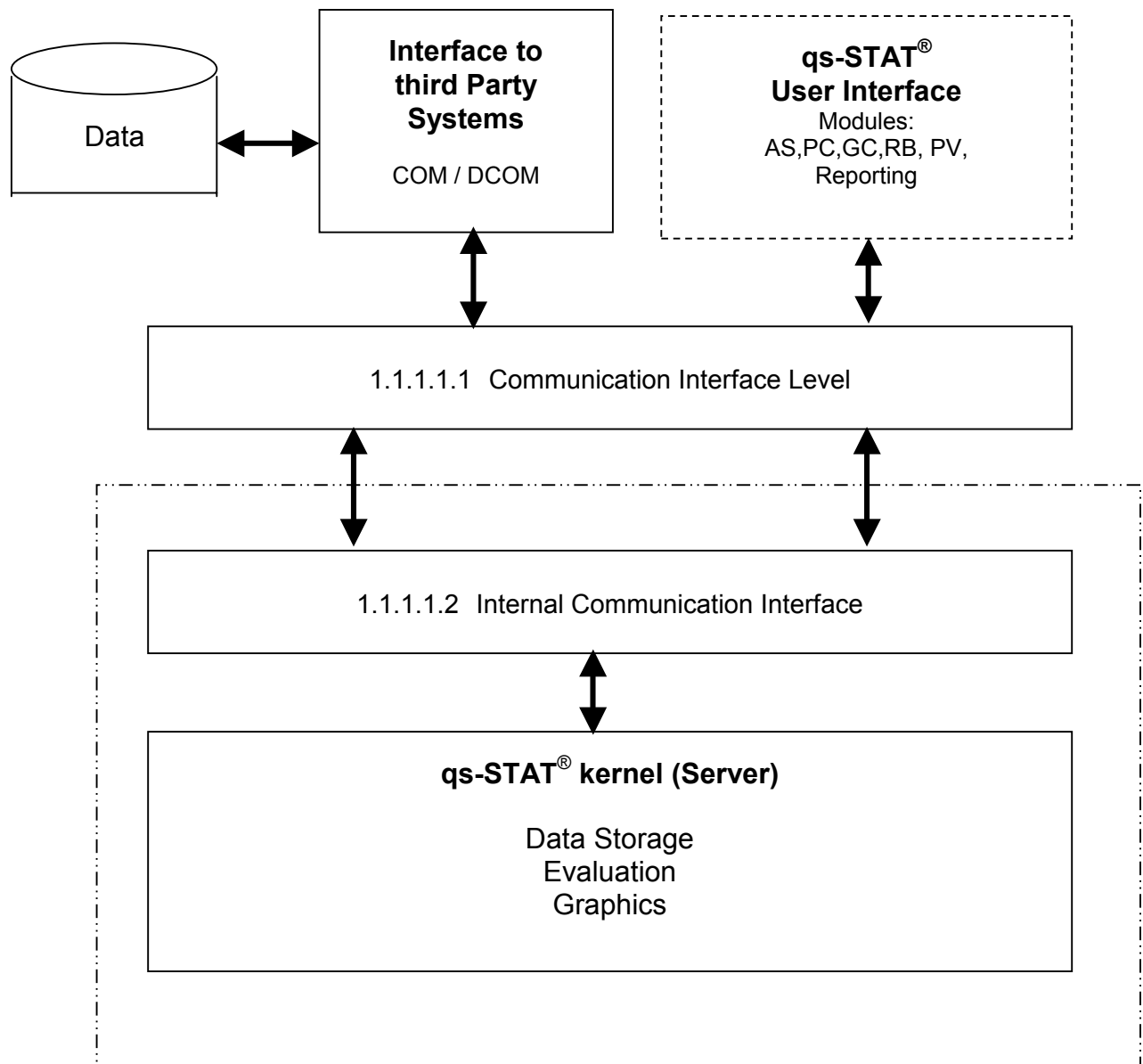


# **Integration of the qs-STAT Statistical Server into 3<sup>rd</sup> Party Systems**

<b>1</b>	<b>Overview .....</b>	<b>3</b>
<b>2</b>	<b>Modules .....</b>	<b>4</b>
2.1	Online / Offline Analysis .....	5
2.1.1	General .....	5
2.1.2	Online Server .....	5
2.1.3	Online Analysis Server .....	6
2.1.4	Process capability analysis .....	7
<b>3</b>	<b>COM/DCOM Interface .....</b>	<b>8</b>
3.1	Features of the Q-DAS COM Interface .....	8
3.2	Functionalities of the Q-DAS Server .....	8
3.3	General Communications .....	9
3.4	Processing a Command .....	10
3.5	Informing the Client-Application .....	11
3.6	Watchdog .....	12
3.7	Sending data to the Server .....	13
3.8	Query results and control flow .....	14
<b>4</b>	<b>Connection handling.....</b>	<b>15</b>
4.1	Architecture of multiple connections .....	15
4.2	Events in case of multiple connections .....	16
<b>5</b>	<b>Online Systems .....</b>	<b>17</b>
5.1	Kinds and range of statistical evaluation .....	18
5.2	Example for the usage in pseudo-code .....	19
5.3	Quality control charts for the Online Server .....	21
5.3.1	QCC in the Online Analysis Server .....	22
5.4	Statistical Alarms .....	22
<b>6</b>	<b>Scripting languages .....</b>	<b>23</b>
6.1	Serialized processing of commands .....	23
6.2	Untyped [out] parameters .....	23
6.3	Getting graphical results .....	23

## 1 Overview



## 2 Modules

The Q-DAS Statistical Server is available in two different variations. Mixed usage between these variations is possible. We call these variations “Offline Server”, “Online Server” and “Online Analysis Server” (=Mixed).

Mapped to the qs-STAT modules, the following combinations are possible:

qs-STAT Variation qs-STAT Module	Offline	Online	Online Analysis
Sample Analysis	X	-	-
Process Capability	X	-	X
Procella	-	X	X
Measurement System Analysis	X	-	-

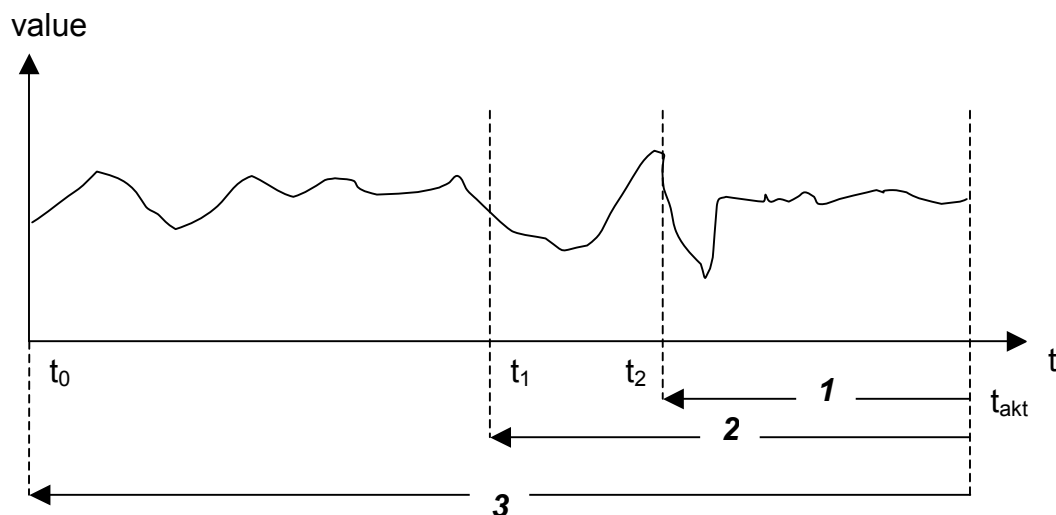
## 2.1 Online / Offline Analysis

### 2.1.1 General

Starting point is a existing data set over a certain period of time. The data is loaded inside the server by either

- Reading form an existing file
- Querying from the Q-DAS database
- Transmitting the data in an memory stream.

After loading the data, the situation is like it follows:



$t_{akt} = \text{now}$

Starting point for the evaluation and the visualization is the timepoint  $t_{akt}$ , which is equivalent to the timepoint of the last existing measurement:

1. The last x values are displayed in a Online System (time period  $t_{akt} - t_2$ ).
2. The last y values could be evaluated in a Online System (time period  $t_{akt} - t_1$ ).
3. A Long term analysis for the time period  $t_{akt} - t_0$  could be implemented in a Offline System.

### 2.1.2 Online Server

In any way data will generally added to existing data sets inside the online server. The online evaluation over the time period  $t_{akt} - t_2$  will be performed every time when a new data set is added to existing ones. This time period is limited through the adjustable number of measurements.

The data itself is stored in a FIFO buffer with adjustable size. It's obvious that the size of the buffer should be larger than range of displayed periode (time period  $t_{akt} - t_1$ ). All measured values, which had fallen out of the buffer periode, are no longer available (-> FIFO strategy).

The following operations could be recalled (and displayed online):

- individual value charts
- bar charts
- QCC
- num. summaries (valuation results, parts protocols, statistics characteristics)
- num. summaries without C-values and test summaries

All time **intense operations**, which have no fixed processing time guaranteed, will not be performed. These are:

- distribution tests
- distribution selections
- C value calculation

## 2.1.3 Online Analysis Server

In principle applies here cap. 3. That means, that a process visualization for the time period  $t_{akt} - t_2$  with the related online evaluation, will be performed.

The extension of the online analysis server exist as follows:

After data was taken over, the a process capability analysis may be done **on request** (e.g. by the user via the client application) over the range of the existing data. Please note, that this operation is **not** done automatically (in opposite to the Online mode, see 2.1.2 Online Server).

The time period for the measured values (which will be analysed) is defined by the adjustable size or rather the adjustable number of values of the FIFO buffer (time period  $t_{akt} - t_1$ ). This a kind of a snapshot, since the calculated statistics will be resetted the next time, when the data changes.

All informations of the process analysis could be recalled. Possibly there are Pp, Ppk values instead of Cp and Cpk values, if the buffer size is selected to small.

### **2.1.4 Process capability analysis**

The whole data stock loaded inside the server will be evaluated on basis of the rules of the process capability analysis (time period  $t_{akt} - t_0$ ).

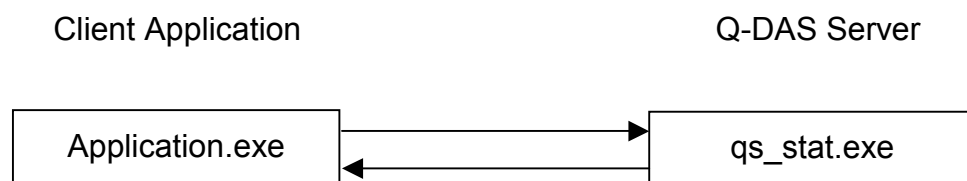
The FIFO buffer is not activated. There are no essential limitations to the amount of data, except from the present RAM and harddisc space and the demands for response time of the system. The necessary resources depend on the total number of characteristics and the number of values of the characteristics with contains maximum of values.

## 3 COM/DCOM Interface

### 3.1 Features of the Q-DAS COM Interface

The Q-DAS interface is implemented as a COM/DCOM component. Here are the most important features of the qs-STAT interface:

- COM/DCOM object
- Implemented as an executable
  - local server
  - remote server
- connectable interface (outgoing interface for giving events to the client application)

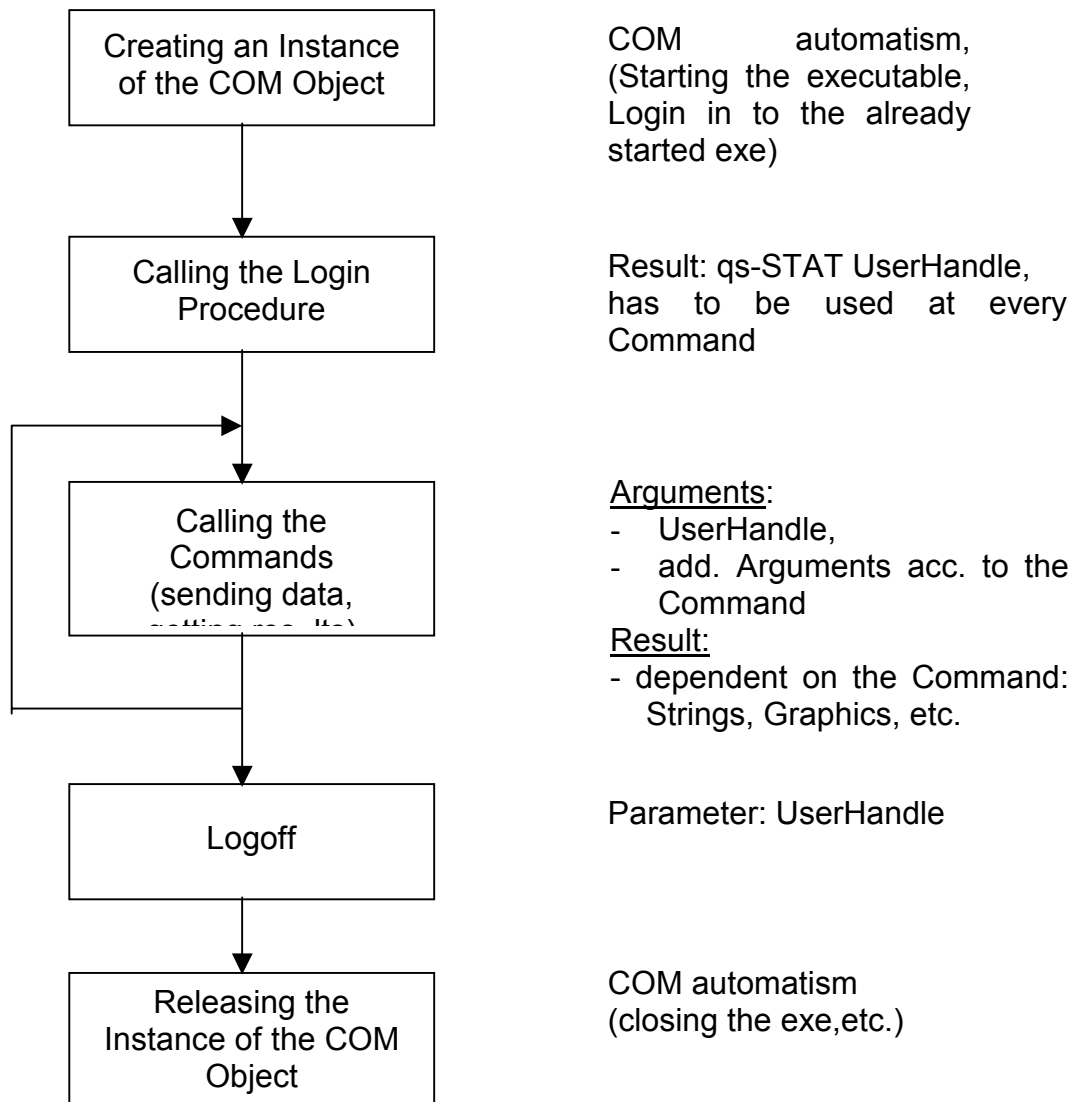


### 3.2 Functionalities of the Q-DAS Server

- statistical evaluation
- data storage
- getting results
  - numerical results
  - graphical results
- getting system information / system settings
  - names of qs-STAT fields
  - contents of qs-STAT fields  
(according to the loaded data: - modules, - languages, - retrieving supported features)
- access to/ querying the Q-DAS database
- Event handling
  - watchdog
  - data synchronization
  - data changes
  - statistical alarms



### 3.3 General Communications



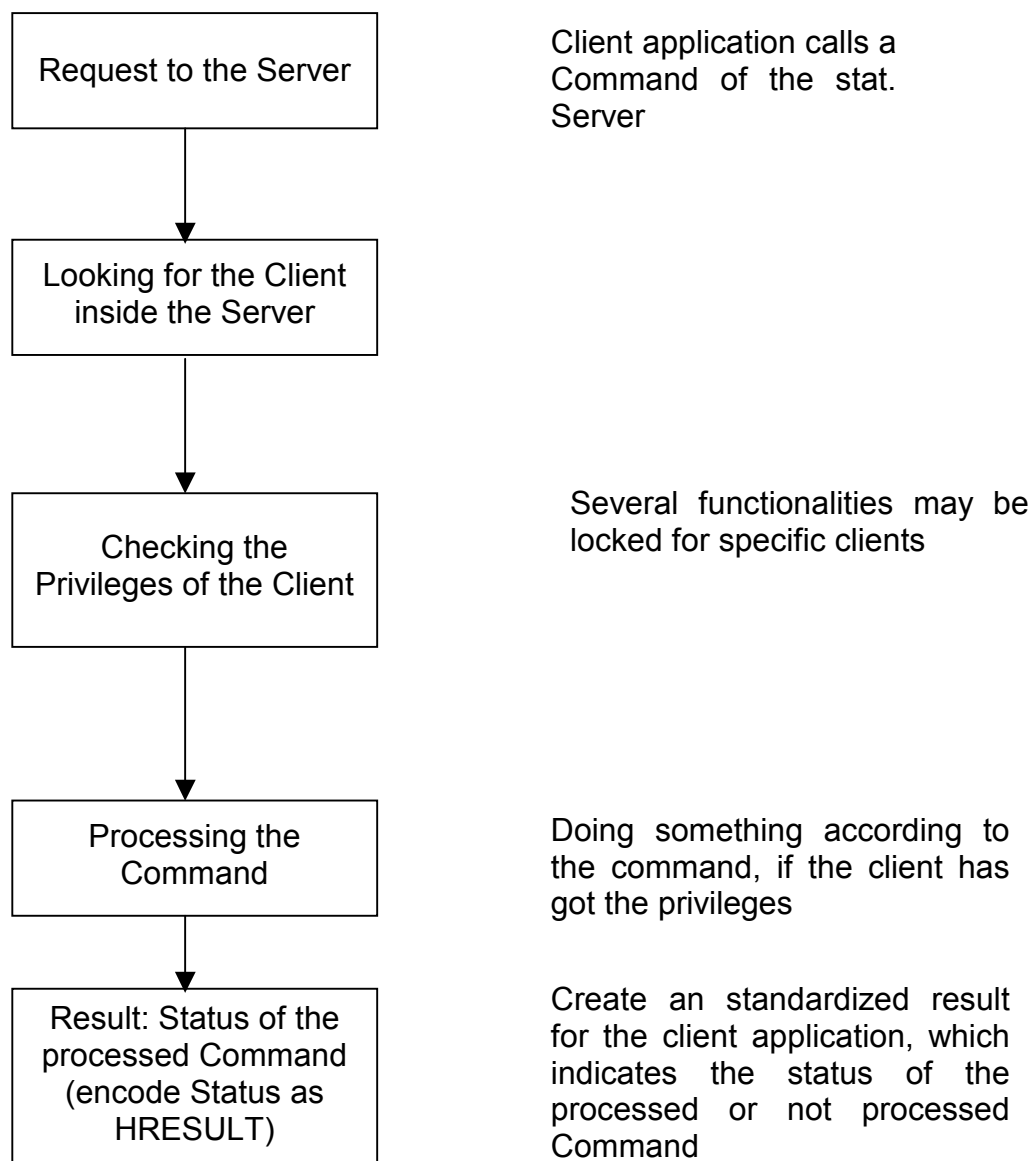
### 3.4 Processing a Command

Arguments:

- UserHandle (stored by the client after an successful login)
- Additional arguments according to the various commands

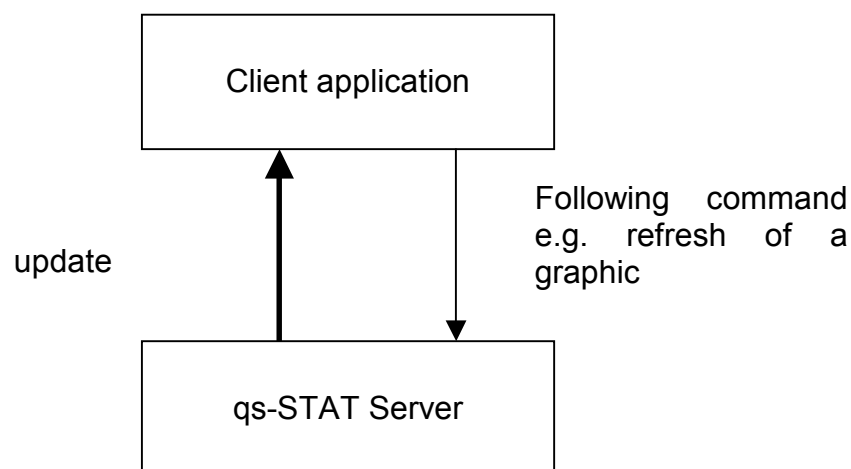
Result:

- success or error code encoded as an HRESULT



### 3.5 Informing the Client-Application

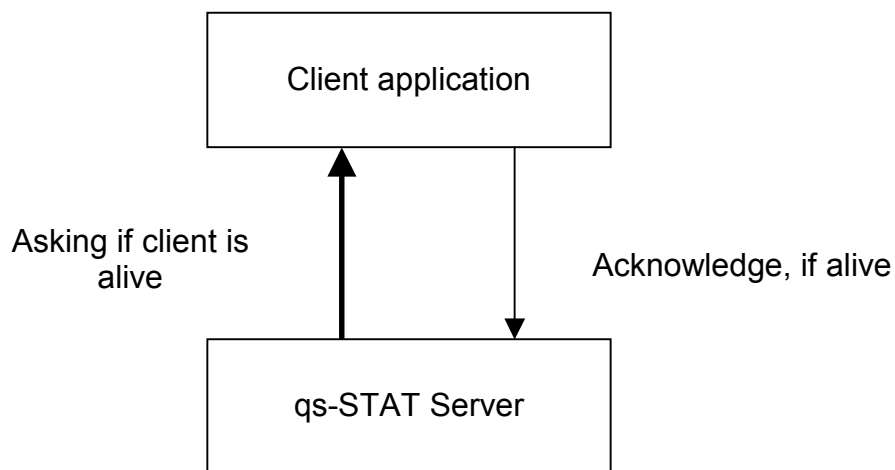
After data is changed or other events, the clients will be informed by the server. This mechanism will be implemented using connection points. The client has to implement this interface, if he wants to get informed about changes of the data or other events created by the qs-STAT server. It is recommended to implement this interface, since several methods of the server (time consuming methods such as loading of data or evaluation the loaded data) run concurrently and a synchronisation of the processes are helpful.



### 3.6 Watchdog

The Q-DAS server asks in cyclic intervals, if the client application is alive. If the Client doesn't respond to the request within a certain time, the client is removed from the system.

This mechanism is done by the connection point technology. The client application has to provide this interface. If not, no communication will be established.



### 3.7 Sending data to the Server

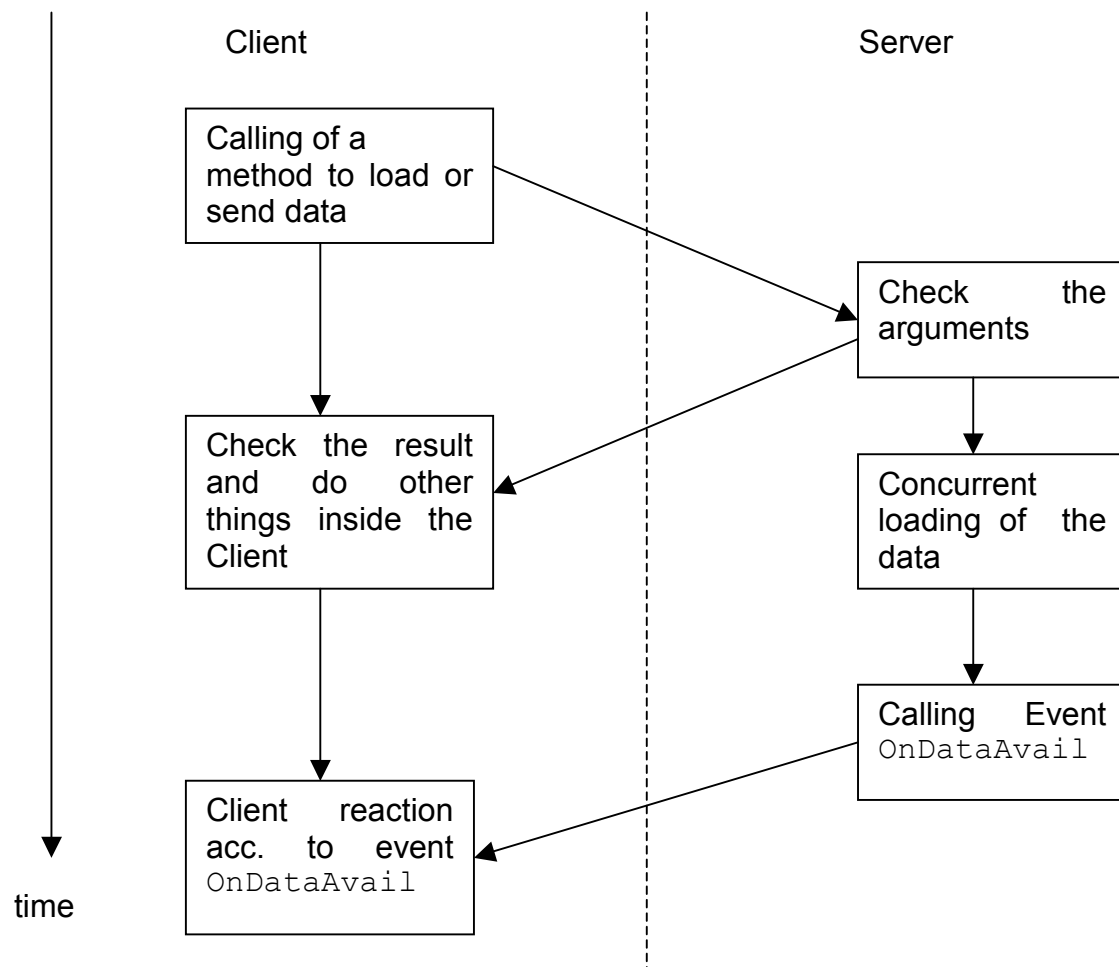
The data processed by the statistical server, has to be send in the Q-DAS ASCII-data format.

The servers accepts both,

- files or
- memory streams.

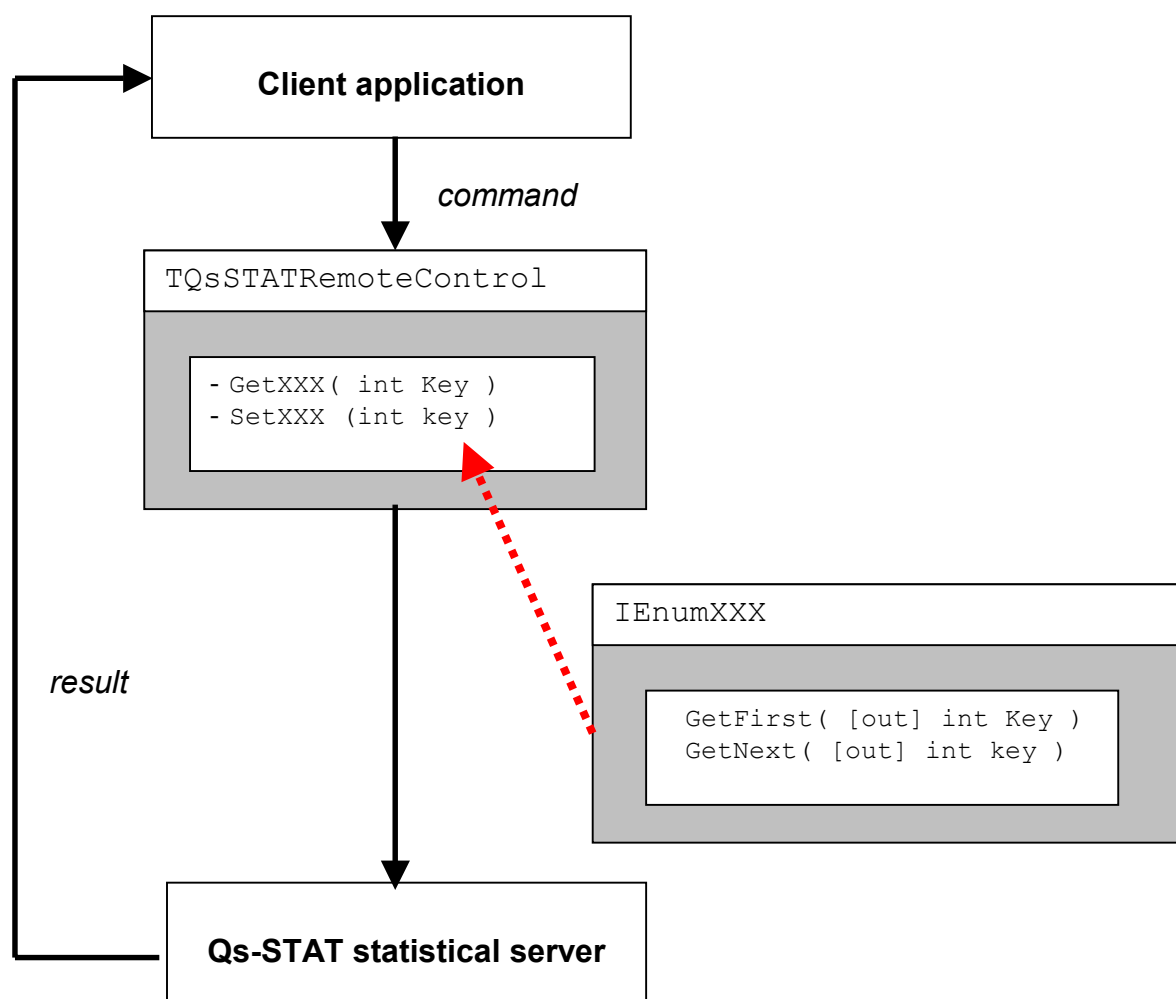
In online systems, single DFX-lines or blocks of DFX lines may be send to the server, which will be added to a loaded set data. After an evaluation, the new statistics may be called by the client application. For larger amount of data, the DFX or DFD/DFX files may be split in smaller portions. This is recommended especially for the transmission of data via memory stream.

The loading of the data is done concurrently by the server. When the data is loaded, the client will be informed through the connection point interface.



### 3.8 Query results and control flow

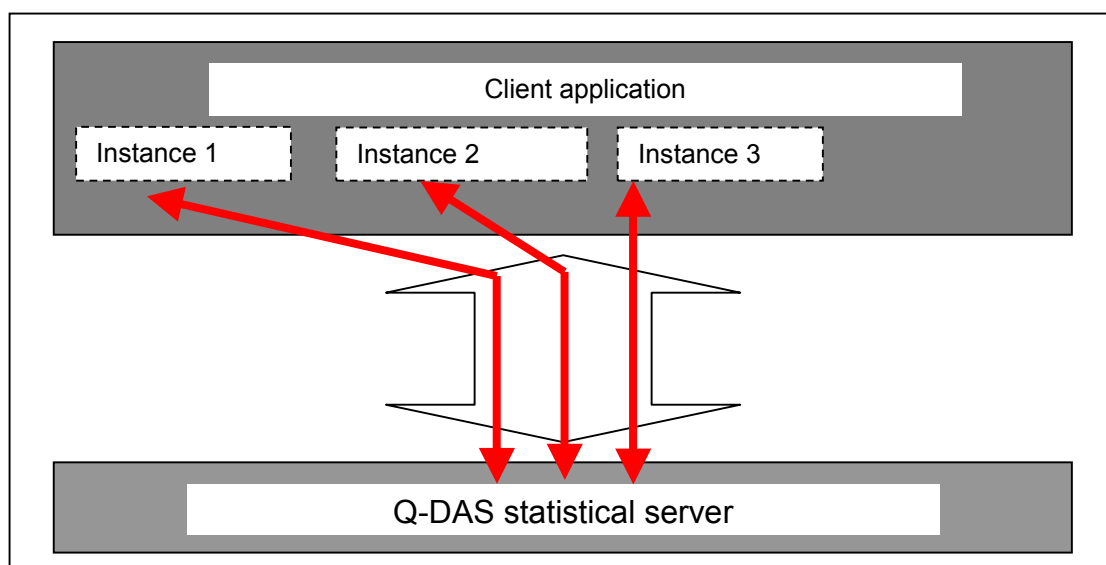
To reduce the total number of server methods, the commands are unified for certain groups of commands (e.g. for getting something from the server or for setting something at the server). The specification of these server methods is done by a parameter of the method. Valid values for these parameters can be obtained by a call to the corresponding Enumeration interfaces (e.g. `IEnumLanguage`, `IEnumGraphics`).



## 4 Connection handling

### 4.1 Architecture of multiple connections

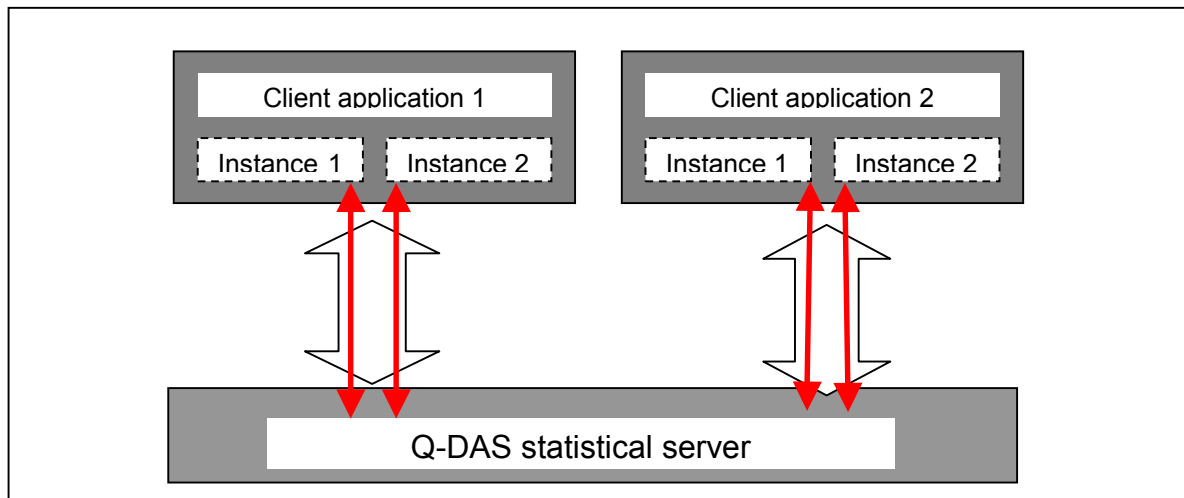
Each client might use one or multiple connections to the same time. This can be done by several calls to the connect commands. The client retrieves a unique handle for each connection, that has to be stored on the client side for having an access to certain connections. Each connection represents a “data instance” on the server side, while each “data instance” runs independently from the others (modules, languages, loaded data,...).



*One client uses multiple connections*

The server supports multiple clients at the same time. To use the server from an instance of a client application, the client has to create a connection to the server.

**This method can be used to load and evaluate several data sets at the server.**



*Multiple clients use multiple connections*

If a certain connection is no more in use, the client can cancel this connection by the disconnect commands.

## 4.2 Events in case of multiple connections

In case of concurrently processed commands, events are sent to client applications to offer a possibility of synchronization. The events (defined as connection points) are sent to the client applications directly and cover all established connections. To distinguish between events of different connections, the connection handle, that caused the event, is defined in the methods of the event interface. The client has to keep track of its connection handles and has to separate the events on its own by a comparison of the stored and the transmitted (by the events) connection handles .

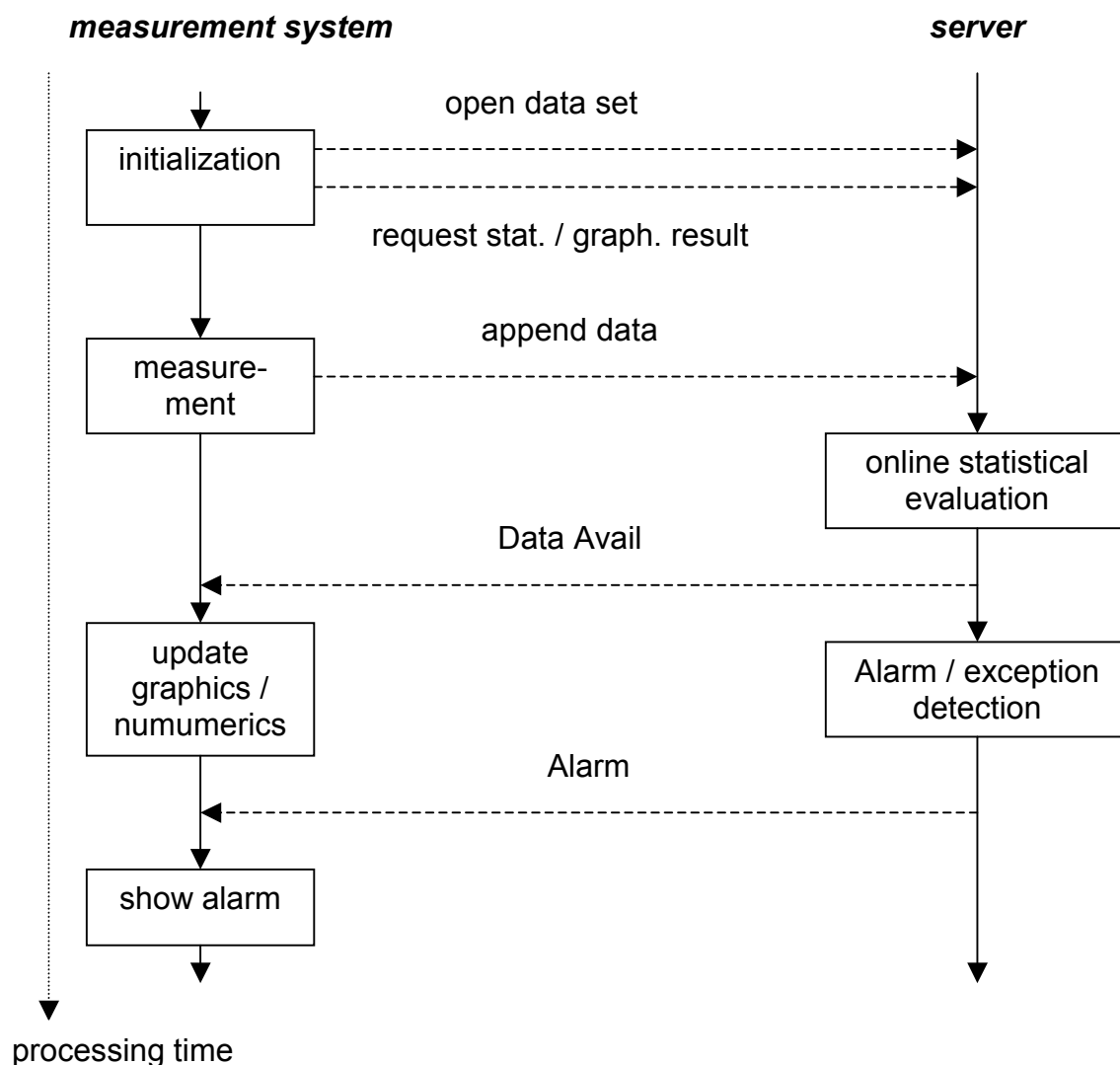


## 5 Online Systems

Generally the online interface supports the same methods as the offline interface does. In addition, the online interface does a view things automatically. Also the alarm (exception) interface is activated and the clients will be informed according to alarm conditions after a data change.

Please take note, that the “online-statistics” is a subset of the “offline statistic” and not all statistical results are available for online systems. The online statistical evaluation is based on a predefined subset of the data present in the system ( see 2.1 Online / Offline Analysis).

Below there is a typical scenario, how the communications between client and server maybe done:



## 5.1 Kinds and range of statistical evaluation

Basically there are 2 different kinds of evaluations (see 2.1 Online / Offline Analysis).

The evaluation of the Online Sever (which is done automatically after the dataset has changed) is doing a reduced process capability evaluation (without finding a suitable distribution model and also without all statistics, which is based on a certain distribution model). The range of values of this evaluation is derived from the range for the graphical display. The number of values that is displayed, is a system setting of the server and can be changed there.

In this case only a subset of the statistics and the graphics are available.

The Online analysis server does in addition a process capability evaluation on **user request**. This means, that the client application can perform a call to the

`TQsSTATRemoteControl.EvaluateXXXX` – methods,

and the data inside the FIFO buffer is evaluated according the the definition of the presetted (by a call to the `TQsSTATRemoteControl.SetEvaluationStrategy` method) of according to the default evaluation strategy. The size of the FIFO buffer and the evaluation strategy as well are system settings of the server and can be changed there.

After a call to the `TQsSTATRemoteControl.EvaluateXXX-methods` all statistics and graphics of the process capability analysis are available (as a snapshot of the data) until the data will be changed the next time.

## 5.2 Example for the usage in pseudo-code

This example contains of 3 procedures, which do the most important interactions for getting a graphic (value chart) online and for doing an process capability study for the data available inside the FIFO buffer, including an snapshot of the C-Value plot.

This could be used in a kind of sequence:

```
procedure Main()
    OLEVARIANT vaData;      // for transmission of Data in
                           // Ascii-Transferformat

    // .some code for getting the existing data and formatting it
    // acc. to specs of Q-DAS Ascii-Transferformat
    ...

    // data is in vaData now
    // get a startup graphic
    OpenBaseDataForOnlineVisualisation(vaData)

    // measuring cycle, retrieve new data from the gage
    // Format data and store it inside the OLEVARIANT
    ...

    // new data is formatet now, get a graphic for online-Display
    AppendMeasuredDataAndRefreshGraphic(vaData)

    // finally do a process capability analysis for the FIFO buffer
    // and get a C-value plot
    ProcessCapabilityOnRequestAndCValuePlot()

end // of main proc.
```

// data acquisition / data collection phase:  
// data is stored inside of an OLEVARIANT in form of Q-DAS Ascii-Transferformat  
// data contains part, characteristic and value informations

```
procedure OpenBaseDataForOnlineVisualisation( OLEVARIANT va)
    OLEVARIANT vaGraphicInfo; // for graphic transmission

    // transmit data to the server and evaluate automatically
    QSSTATRemoteControl.TransmitFileExt( va,0,0)

    // waiting until data is processed inside the server
    WaitForSingleObject(DataAvailEvent, timeout)

    // Request for a graphic (value plot = 3100)
    QSSTATRemoteControl.GetGraphicExt(3100,vaGraphicInfo)

    // draw the retrieved graphic on an HDC
    SCREEN.HDC.Draw(vaGraphicInfo)
end //of procedure
```

```
// append data which was collected
// collected data is stored inside of an OLEVARIANT in form of Q-DAS
// Ascii-Transferformat
// data contains value informations only
```

## **procedure AppendMeasuredDataAndRefreshGraphic( OLEVARIANT va)**

```
OLEVARIANT vaGraphicInfo; // for graphic transmission

// transmit data to the server and evaluate automatically
// use append data mode
QSSTATRemoteControl.TransmitFileExt( va,2,0)

// waiting until data is processed inside the server
WaitForSingleObject(DataAvailEvent, timeout)

// Request for a graphic (value plot = 3100) containing new data
QSSTATRemoteControl.GetGraphicExt(3100,vaGraphicInfo)

// draw the retrieved graphic on an HDC
SCREEN.HDC.Draw(vaGraphicInfo)
end //of procedure
```

```
// preset an evaluation strategy and evaluate the data stored inside of the
// FIFO buffer
// show a C-value plot on the screen
```

## **procedure ProcessCapabilityOnRequestAndCValuePlot()**

```
OLEVARIANT vaGraphicInfo; // for graphic transmission

// select the an existing evaluation strategy (=1)
QSSTATRemoteControl.SetEvaluationStrategy(1)

// evaluate the data inside the FIFO buffer acc. to process
// capability
QSSTATRemoteControl.EvaluateAllChars()

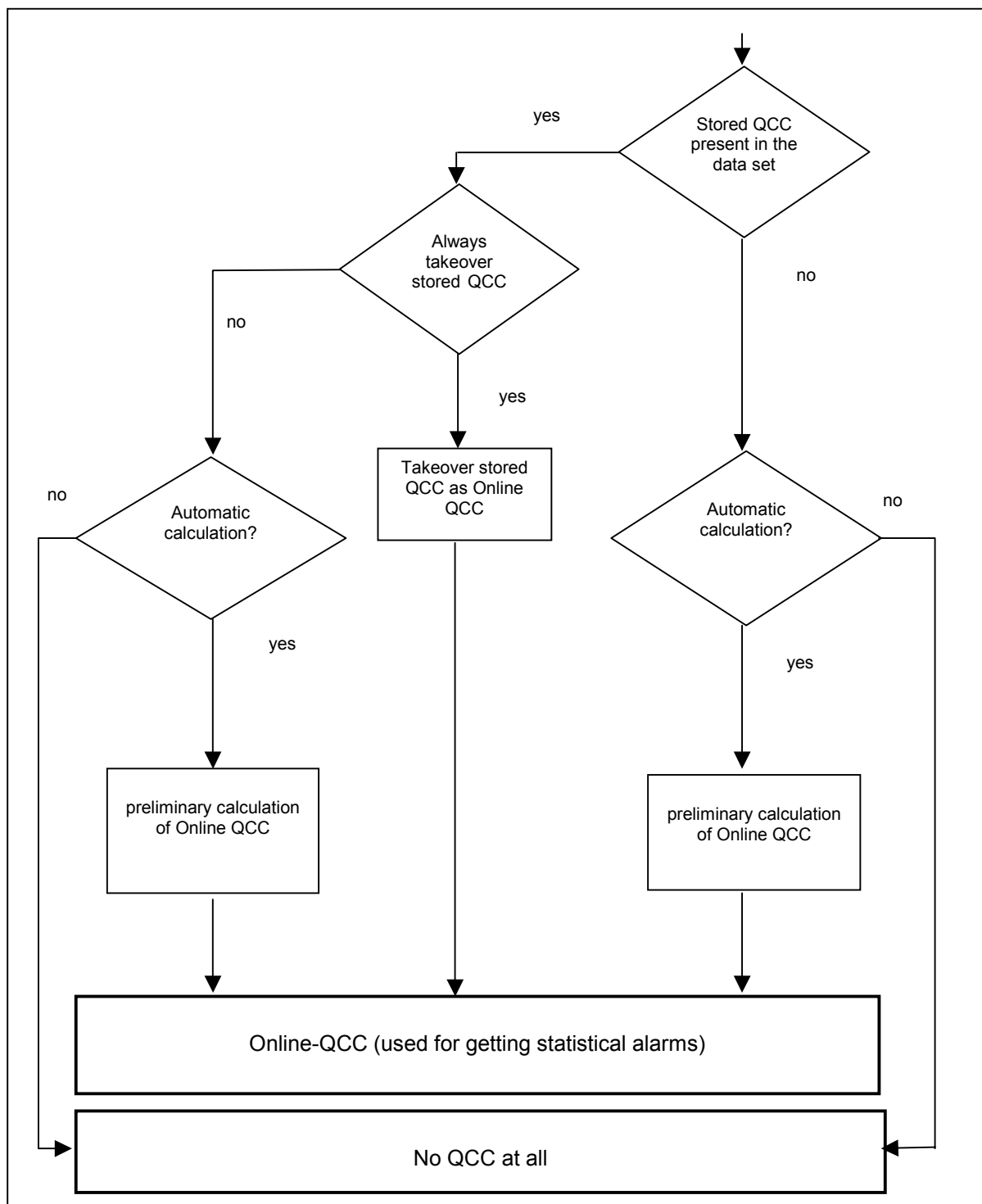
// waiting until data is evaluated inside the server
WaitForSingleObject(EvaluationDoneEvent, timeout)

// Request for a graphic (C-value plot = 7200) containing
// data inside the FIFO buffer
QSSTATRemoteControl.GetGraphicExt(7200,vaGraphicInfo)

// draw the retrieved graphic on an HDC
SCREEN.HDC.Draw(vaGraphicInfo)
end //of procedure
```

### 5.3 Quality control charts for the Online Server

The Online server uses a QCC that is called the “Online QCC” for the detection of statistical alarms. There are several ways to get this Online QCC, which are described in the following flow chart. Which track is used after a data set is loaded and evaluated, is defined in the evaluation strategy.



Getting the Online QCC is done automatically after a data set is loaded in the Online Server (Online analysis server) according to the presetted evaluation strategy. There are possibilities to:

- use that QCC, which is defined in the data set
- calculate a preliminary QCC
- don't use a QCC

Note, that the usage of this QCC has no effect on the data set itself, but a client can get the current QCC parameters by a call to the `GetStatResultExt` – method (result key = 8xxx, Subkey = 4) to store these parameters in the data set.

As a result of a certain obtained Online QCC, the alarms are created and notified to the client by the `IQsEvent` – Sink interface. This happens, when data has been changed.

## 5.3.1 QCC in the Online Analysis Server

In addition to 5.3.1 , when a process capability over the range of the FIFO buffer is done by calling the `EvaluateXXXX` methods, there is a possibility to takeover a QCC, which limits are calculated on a more detailed way, according to the rules for the process capability analysis. The QCC parameters are taken from a QCC, which is called the “SPC-QCC”. How the parameters for this QCC are calculated is defined also in the evaluation strategy. The settings allow you the following possibilities:

- Taking over the newly calculated SPC - QCC for all characteristics automatically
- Taking over the newly calculated SPC - QCC for all characteristics automatically, if there are no stability warnings
- No take over of the SPC - QCC

Note, that the usage of this QCC has no effect on the data set itself, but a client can get the current QCC parameters by a call to the `GetStatResultExt` – method (result key = 8xxx, Subkey = 3) to store these parameters in the data set.

## 5.4 Statistical Alarms

The **Online Server / Online Analysis Server** detects statistical alarms after a change of the data. In case of an alarm condition, the encoded statistical alarms will be sent to the connected client application through the `IQsEvent` Interface. Client applications can use this information to inform the user about the alarm conditions. Alarms are raised according to the “Online QCC” ( see 5.3). The alarm setup itself is stored at the each evaluation strategy individually. For more details, please refer to the qs-STAT manuals.

## 6 Scripting languages

Scripting languages like VB Script, which cannot use typed variables and cannot react to events, have also an access to the qs-STAT server.

### 6.1 Serialized processing of commands

All methods which are designed for concurrent processing and which need a synchronisation (done by the Event-interface) can switch the concurrent processing off. Therefore exists a number of methods, which have the extension **TM** at the end of the method name ( e.g. `EvaluateAllCharsExt` ). These functions have the same parameters as their concurrent equivalents including an additional parameter named `ThreadMode`, which can be used to switch the concurrent processing off.

### 6.2 Untyped [out] parameters

A lot of methods have output parameters to pass results to client applications. In general these parameters are typed parameters (for. Example as a double or string value).

All these methods have equivalents, that pass the values in kind of an OLEVARIANT. They have the extension **VA** in their method name. All other parameters have the same meaning as their equivalent methods.

### 6.3 Getting graphical results

To get graphical results, these languages may use functions, that have the extension **"AsFile"**. They pass the name of an temporary file back to the client, which contains the requested graphic in the requested graphic format.

Example: `TqsstatRemoteControl.GetGraphicExtAsFile` creates a temporary file on the local hard-disk that contains the graphic and returns the full file name of that file to the client.