SECTION 18

Table Definitions

IN THIS SECTION

18.1 Configuration tables

A configuration table is maintained in which every group in the historian is stored along with the source of its data. Each group is assigned a unique history ID. The history IDs are not recycled, which ensures that you can always access historical data.

TABLE NAME:- CFG_GROUP	
COLUMN NAME	**COLUMN DESCRIPTION**
TYPE	Scan Group
NAME	ASCII config group name
ID	Unique integer ID
SOURCEID	
SCANNERID	ID of originating scanner for a point group
SCAN_MSECS	Frequency of scanning in milliseconds
DB_VAL	Compression deadband value
DB_K_VAL	Compression deadband additional information
DB_N_VAL	Compression deadband additional information
CONDITIONID	
MAX_SAVE_TIME	
DESCRIPTION	English description
ENABLED	Point group enables status: Y or N
DB_ALGNAME	Algorithm name
QUALIFIER	

A configuration table is maintained in which every point name in the historian is stored along with the source of its data. Each point is assigned a unique history ID. The history IDs are not recycled, which ensures that you can always access historical data. Point names are never deleted from this table, though you can modify the point information.

The same history ID is retained permanently for a specific point name, though you can modify other point information because it does not compromise data retrieval. This history ID provides the key to store and retrieve data to/from the current, historical and modified tables. Inactive points can be marked as disabled because they are never deleted from the table. The type of data that is associated with each point is configured so that collection, storage and retrieval can handle the data in the historical data tables.

TABLE NAME:- CFG_PT_NAME	
COLUMN NAME	COLUMN DESCRIPTION
NAME	ASCII point name
ID	Unique integer ID
PT_FLAGS	Flags for processing options
SOURCE_ID	ID of originating data scanner/source
SOURCE_NAME	Name of point in originating system
CATEGORY	For example, Temp, Pressure, Boiler, etc.
IMPORTED	Indicates if point is imported
ENABLED	Indicates if point is currently enabled
DATA_TYPE	Analog ('R'), Digital ('D'), Packed ('P')
INVERTED	Indicates if point values are inverted. Not applicable.
SCAN_MSECS	Frequency of scanning in milliseconds (100 msec or multiple of 1000)
DB_ALGCODE	Compression type, deadband information, etc.
DB_ALGNAME	Compression type, deadband information, etc.
DB_VAL	Compression deadband
DB_K_VAL	Compression deadband additional information
DB_N_VAL	Compression deadband additional information
SCAN_GROUP_ID	ID of assigned scan group
PT_INFO	Contains DCS specific point parameters
CREATION_TIME	Date and Time point was added
MODIFY_TIME	Date and Time point was last modified
MAX_SAVE_TIME	
SOURCE_TYPE	Identifies as a scanned point, lab point, or statistics point.

A configuration table is maintained in which every point name in the historian is stored along with the source of its data. Each point is assigned a unique history ID. The history IDs are not recycled, which ensures that you can always access historical data. Point names are never deleted from this table, though you can modify the point information.

The same history ID is retained permanently for a specific point name, though you can modify other point information because it does not compromise data retrieval. This history ID provides the key to store and retrieve data to/from the current, historical and modified tables. Inactive points can be marked as disabled because they are never deleted from the table. The type of data that is associated with each point is configured so that collection, storage and retrieval can handle the data in the historical data tables.

Table Name:- CFG_PT_NAME	
Column Name	**Column Description**
NAME	ASCII point name
ID	Unique integer ID
PT_FLAGS	Flags for processing options
SOURCE_ID	ID of originating data scanner/source
SOURCE_NAME	Name of point in originating system
CATEGORY	For example, Temp, Pressure, Boiler, etc.
IMPORTED	Indicates if point is imported
ENABLED	Indicates if point is currently enabled
DATA_TYPE	Analog ('R'), Digital ('D'), Packed ('P')
INVERTED	Indicates if point values are inverted. Not applicable.
SCAN_MSECS	Frequency of scanning in milliseconds (100 msec or multiple of 1000)
DB_ALGCODE	Compression type, deadband information, etc.
DB_ALGNAME	Compression type, deadband information, etc.
DB_VAL	Compression deadband
DB_K_VAL	Compression deadband additional information
DB_N_VAL	Compression deadband additional information
SCAN_GROUP_ID	ID of assigned scan group
PT_INFO	Contains DCS specific point parameters
CREATION_TIME	Date and Time point was added
MODIFY_TIME	Date and Time point was last modified
MAX_SAVE_TIME	
SOURCE_TYPE	Identifies as a scanned point, lab point, or statistics point.

A configuration table is maintained in which every scanner in the historian is stored along with the source of its data. Each scanner is assigned a unique history ID. The history IDs are not recycled, which ensures that you can always access historical data.

Table Name:- CFG_DB_SOURCE	
Column Name	**Column Description**
NAME	Unique name for the data source
ID	Unique Source ID
PARTNER_ID	ID of configured partner scanner
TYPE	Type of data source, for example, Pt Scanner, Attribute Scan
DOWNLOAD_TIME_A	Time of last download for point scanner
DOWNLOAD_TIME_B	Time of last download for attribute scanner
NET_ADDRESS	Character string that contains the TCP/IP address for scanner workstation.
ENABLED	Source drop status

18.2 Point history tables

Each data sample that is generated is stored in the historical data tables. These tables contain timestamped value and status information for the points that are being collected by the historian server. Entries are stored in the history table at a fixed minimum frequency, even when no changes occur. This enables you to determine the value of a point at the beginning of your requested retrieval interval. The fixed minimum frequency is set at one hour. The frequency may be longer for migrated data.

Status data is stored in a normalized format where common status information is recorded for all points, regardless of their source. The DCS scanner process performs normalization on status information, value formats, and time formats prior to sending data to the historian server collection process.

This table shows how the status information is normalized from an Ovation System.

Status Definition		Point Type	Mapped Emerson Ovation Status	
Bit 0	Not reset/Alarm reset bit	LA, LD, LP	Status 1	Bit 19
Bit 1	Tagged Out	LA, LD, LP	Status 3	Bit 2
Bit 2	Low Limit Exceeded	LA	Status 1	Bit 2
Bit 3	High Limit Exceeded	LA	Status 1	Bit 3
Bit 4	Hardware Error	LA, LD, LP	Status 3	Bit 3
Bit 5	Alarm Unacknowledged	LA, LD, LP	Status 1	Bit 5

Status Definition		Point Type	Mapped Emerson Ovation Status	
Bit 6	Cut out from Alarm Checking	LA, LD, LP	Status 1	Bit 6
Bit 7	Point is in Alarm	LA, LD, LP	Status 1	Bit 7
Bit 8	Quality	LA, LD, LP	Status 1	Bit 8
Bit 9	Quality	LA, LD, LP	Status 1	Bit 9
Bit 10	Current Value is operator entered	LA, LD	Status 1	Bit 10
Bit 11	Point is removed from scan/forced	LA, LD, LP	Status 1	Bit 11
Bit 12	Limit Checking off	LA	Status 1	Bit 12
Bit 13	Alarm Checking off	LA, LD, LP	Status 1	Bit 13
Bit 14	Oscillating Point Low UDA	LD, LP LA	Status 3 Status 1	Bit 4 Bit 22
Bit 15	Point is timed out	LA, LD, LP	Status 1	Bit 15
Bit 16	Analog Limit Number (Better Bit if bit 18 is set)	LA	Status 1	Bit 16
Bit 17	Analog Limit Number (Worse Bit if bit 18 is set)	LA	Status 1	Bit 17
Bit 18	Analog/Packed Incremental Bit	LA, LP	Status 1	Bit 18
Bit 19	Uncommissioned Bit	LA, LD, LP	Status 3	Bit 15
Bit 20	SID Limit Error	LA, LD, LP	Status 1	Bit 20
Bit 21	UDA Limit Packed Device	LA LP	Status 1 Status 1	Bit 21 and Bit 22 Bit 21
Bit 22	Quality Latched	LA	Status 2	Bit 14

Note: If both bits 2 and 3 are set, the point is in Sensor Alarm. If point is an analog and bit 21 is set, then the point is in UDA alarm. If bit 14 is set, then the point is in low UDA. If bit 14 is reset, then the point is in high UDA.

Table Name:- PT_HF_HIST	
Column Name	Column Description
ID	Unique point ID

Table Name:- PT_HF_HIST	
TIMESTAMP	Timestamp in UTC
TIME_NSEC	High resolution time stamp in nanoseconds
SAMP_FLAGS	Indicates duplicate data, missing data, and modified data
F_VALUE	Floating Point value of sample, as appropriate, for analog points
RAW_VALUE	Raw value of sample, as appropriate, for packed points
STS	Status and Quality indicators for value fields
SAMP_CREATE_TIME	Timestamp that the sample is inserted
SAMP_CREATE_SEQ	Sequence number on the time that the sample is inserted

Table Name:- PT_ATTRIB	
Column Name	Column Description
ID	Unique point ID
TIMESTAMP	Timestamp in UTC
TIME_NSEC	High resolution timestamp in nanoseconds
SAMP_FLAGS	Flag information
DESCRIPTION	English Description
AUX_DESCRIPTION	Auxiliary Description
ENGINEERING_UNITS	Engineering Units description
SET_DESCRIPTION	Set state description for digital points
RESET_DESCRIPTIION	Reset state description for digital points
SIGNIFICANT_DIGITS	Number of decimal places to display
BOTTOM_SCALE	Bottom of scale value for display
TOP_SCALE	Top of scale value for display
LOW_LIMIT	Low limit for alarming purposes
HIGH_LIMIT	High limit for alarming purpose

Table Name:- PT_HF_CURR	
Column Name	Column Description
ID	Unique point ID
TIMESTAMP	Timestamp in UTC
TIME_NSEC	High resolution timestamp in nanoseconds

TABLE NAME:- PT_HF_CURR	
SAMP_FLAGS	Indicates duplicate data, missing data, and modified data
F_VALUE	Floating Point value of sample, as appropriate, for analog points
RAW_VALUE	Raw value of sample, as appropriate, for packed points
STS	Status and Quality indicators for value fields
SAMP_CREATE_TIME	Timestamp that the sample is inserted
SAMP_CREATE_SEQ	Sequence number on the time that'the sample is inserted

TABLE NAME:- PT_LAB_HIST	
COLUMN NAME	COLUMN DESCRIPTION
ID	Unique point ID
TIMESTAMP	Timestamp in UTC
TIME_NSEC	High resolution timestamp in nanoseconds
SAMP_FLAGS	Indicates duplicate data, missing data, and modified data
STS	Status and Quality indicators for value fields
F_VALUE	Floating Point value of sample, as appropriate, for analog points
MEMO	Optional annotation on lab data sample
SAMP_CREATE_TIME	Time at which a sample is inserted into historian
SAMP_CREATE_SEQ	Sequence number on the time that the sample is inserted

18.3 Message history tables

The historian can store messages that originate from the DCS scanners, as well as messages that originate locally due to historian events. Multiple message types are supported, including:

- DCS data messages such as alarms, operator events, ASCII messages.
- DCS event messages such as time shifts or missing data.
- Link messages such as link establish and failure messages.
- Historian event messages such as disk space errors.

TABLE NAME:- MSG_ALARM_HIST	
COLUMN NAME	COLUMN DESCRIPTION
TIMESTAMP	Timestamp in UTC
TIME_NSEC	High resolution time stamp in nanoseconds
MSG_FLAGS	Message flags

TABLE NAME:- MSG_ALARM_HIST	
MSG_TYPE	Message type of alarm, for example: cutout, timeout, high, low, return. For additional information, refer to *Ovation Operator Station User Guide*.
SUB_TYPE	Subtype for alarm filtering
SOURCE_ID	ID of scanner which sent the message
SYSTEM	ID of the system which originated the message
NODE	Node or drop that originated the message
ALM_NAME	The name of the point for which an alarm occurred
PRIM_TEXT	200 character alarm text message storage
ALARM_INFO1	Additional message information

Alarm Subtypes

SUBTYPE NUMBER	DEFINITION
0	No entry
1	New Alarm
2	Return
3	Incremental Alarm
4	Status Change
5	Timed Out Drop
6	Spurious Alarm
7	Suppressed Alarm
8	Acknowledged Alarm
9	Released Alarm

TABLE NAME:- MSG_SOE_HIST	
COLUMN NAME	COLUMN DESCRIPTION
TIMESTAMP	Timestamp in UTC
TIME_NSEC	High resolution time stamp in nanoseconds
MSG_FLAGS	Bit flags
SOURCE_ID	ID of scanner which sent the message
SYSTEM	ID of the system which originated the message
NODE	Node or drop that originated the message
PRIM_TEXT	200 character SOE message storage

TABLE NAME:- MSG_SOE_HIST	
SOE_INFO1	Additional message information

TABLE NAME:- MSG_TEXT_HIST	
COLUMN NAME	COLUMN DESCRIPTION
TIMESTAMP	Timestamp in UTC
TIME_NSEC	High resolution time stamp in nanoseconds
MSG_FLAGS	Message flags
MSG_TYPE	Text message type, for example, Opevent, System, etc.
SUB_TYPE	Sub type category for additional filtering
SOURCE_ID	ID of scanner which sent the message
SYSTEM	ID of the system which originated the message
NODE	Node or drop that originated the message
PRIM_TEXT	200 character text message storage
SUPP_TEXT	100 character optional supplementary message text
SUPP_INFO1	Additional message information
SUPP_INFO2	Additional message information

18.4 Archive volume tables

Each entry in the ARC_VOLUME_LIST table corresponds to one side of your removable media. You are prompted to label each side or removable media when it is created. The label is comprised of three fields from this table (VOLDB:SET:NUM), so that a volume label might look like this: 61:1:0.

TABLE NAME:- ARC_DEVICE_LIST	
COLUMN NAME	COLUMN DESCRIPTION
TYPE	Drive type (for example, DVDRAM)
NAME	Drive name
WIN_DRIVE	Drive letter
STATE	Drive state (for example, IDLE or REQUEST)
FLAGS	Flag information
VOLDB	Volume database name
SET	Volume set
NUM	Volume number

TABLE NAME:- ARC_DEVICE_LIST	
OPEN_TIME	Date and time that the volume was opened
MEDIA_TYPE	Media type that contains the volume

TABLE NAME:- ARC_VOLUME_LIST	
COLUMN NAME	COLUMN DESCRIPTION
DATABASE_NAME	Name of the volume database, typically the hostname. 25 characters maximum.
SET	Name of the volume set within the volume database.
NUM	Name of the volume number within the volume set.
MEDIA_TYPE	Type of media that contains the volume: DVDRAM.
FREE_SPACE	Number of bytes available on the volume.
OPEN_TIME	Date and time that the volume was opened.
CLOSE_TIME	Date and time that the volume was closed. If null, the volume is open. If Close Time is completed, the volume is full.

TABLE NAME:- ARC_VOLUME_DATA	
COLUMN NAME	COLUMN DESCRIPTION
SET	Name of the volume set.
NUM	Name of the volume number.
DATA_TYPE	Type of data: HF, Alarm, SOE, Text, Pt Attr, or Lab.
START_TIME	Earliest data timestamp on the volume for the data type.
END_TIME	Most recent data timestamp on the volume for the data type.

18.5 History Edit and Annotation tables

The annotations that you make using the history edit tool become part of your historical data, and are archived and managed like the rest of you historical data.

These are the annotation types that can be queried through SQL:

TABLE NAME:- ANNOTATION	
COLUMN NAME	COLUMN DESCRIPTION
S_TIME	Start timestamp in UTC
NSECS	High resolution start timestamp in nanoseconds
HISTORYTYPE	Type of historical data (HF, Attr, EAL, etc)
DATATYPE	Annotation type (Main or XREF)
SEQUENCE	Sequence number
SAMP_FLAGS	Flag information
SAMP_CREATE_TIME	Time that the annotation is inserted
SAMP_CREATE_SEQ	Sequence number on the time that the annotation is inserted
E_TIME	End timestamp
E_NSECS	High resolution end timestamp in nanoseconds
DOMAIN	Domain name of where the annotation originates
COMPUTER	Computer name
USERID	User name that inserted the annotation
TEXTSIZE	The length of the Text column's value
TEXT	Annotation text
MAINSEQUENCE	Main sequence number to be cross-referenced
ID	Point ID of the cross-referenced annotation

The edits that have been made to your historical data become part of your historical data, and are archived and managed like the rest of you historical data.

These are edit types that can be queried through SQL:

TABLE NAME:- EDITAUDITLOG	
COLUMN NAME	COLUMN DESCRIPTION
S_TIME	Timestamp in UTC
NSECS	High resolution timestamp in nanoseconds
HISTORYTYPE	Type of historical data (HF, ANN, etc)
SEQUENCE	Sequence number
EDITSTARTTIME	Timestamp when editing started
EDITENDTIME	Timestamp when editing ended
EDITQUALIFIER	ID for HF/LAB history type (TYPE if ANN)
EDITSEQUENCE	Edit sequence number
DOMAIN	Domain name
COMPUTER	Computer name
SAMP_FLAGS	Sample information
USERID	User name

18.6 System tables

The queries in this table enable you to see which retrieval applications are currently connected and using licenses.

TABLE NAME:- SYSLICENSE	
COLUMN NAME	COLUMN DESCRIPTION
THREADID	Thread ID of the connection that's using a license
USERNAME	User name
PASSWORD	Password
APPNAME	Application name
TYPE	Internal or external license
LOGINNAME	User name
DOMAINNAME	Domain name
COMPUTERNAME	Computer name
STATUS	Connection status (Active or Disconnected)
DISCONNECT_TIME	Time of disconnection

18.7 User Defined Function (UDF) tables

The Historian SQL aggregates provide access to interval style processed data through SQL queries. The SQL aggregates pseudo tables utilize User Defined Functions (UDF) syntax to ensure standard compatibility with Excel and Crystal Reports.

The UDF tables are:

- ProcessedData (see page 292)
- ProcDataColWise (see page 296)
- Summary (see page 299)

18.7.1 ProcessedData tables

ProcessedData table

TABLE NAME:- PROCDATA	
COLUMN NAME	COLUMN DESCRIPTION
HANDLE	Point Handle
ID	Points Unique ID
PROCESSING_TYPE	for example, OPH_TIMEAVERAGE, OPH_TOGGLESET
TIMESTAMP	Timestamp in UTC
TIME_NSEC	High resolution time stamp in nanoseconds
SAMP_FLAGS	Flag information
F_VALUE	Floating Point value of sample, as appropriate, for analog points
RAW_VALUE	Raw value of sample, as appropriate, for packed points
TIMESECS	Timestamp in UTC
TIMENSECS	High resolution time stamp in nanoseconds
STS	Status and Quality indicators for value fields
TIMESECS_VALUE	Time value for point as appropriate for example, time of maximum or time set.
TIMENSECS_VALUE	High resolution for point as appropriate for example, time of maximum or time set.
ERROR	

The ProcessedData table provides three syntax options:

- PointsNameOnly (see page 293)
- NameAggrPair (see page 294)
- Complete (see page 295)

PointsNameOnly

PointsNameOnly - where only point names are provided and all points are processed with the same processing type such as, asking for 30 second intervals with TIMEAVERAGE processing over a ten minute period.

Samples

Request time weighted average of three analog points over a 10 minute period specifying IntervalCount of 10 so each interval is one minute.

```
select * from processeddata (

   #06/04/2009 09:00:00#, #06/04/2009 09:10:00#, IntervalCount, 10,

   PointNamesOnly, OPH_TIMEAVERAGE,

   'BLK00101C1A.UNIT2@OV23', 'BLK00101C2A.UNIT2@OV23',
'BLK00101C3A.UNIT2@OV23');
```

Request time weighted average of three analog points over a 10 minute period specifying IntervalSize of 60 seconds so 10 one minute intervals are retrieved.

```
select id, timestamp, f_value, sts from processeddata (

   #06/04/2009 09:00:00#, #06/04/2009 09:10:00#, IntervalSize, 60,

   PointNamesOnly, OPH_TIMEAVERAGE,

   'BLK00101C1A.UNIT2@OV23', 'BLK00101C2A.UNIT2@OV23',
'BLK00101C3A.UNIT2@OV23');
```

NameAggrPair

NameAggrPair - where point names are provided with each having its own specified processing type e.g. asking for 60 * 10 second intervals with each point having a different processing type over a ten minute period

Sample

Request three analog points over a 10 minute period specifying IntervalCount of 10 so each interval is one minute. Request time weighted average for first point, integration for the second and minimum for the third.

```
select * from processeddata (

    #06/04/2009 09:00:00#, #06/04/2009 10:00:00#, IntervalCount, 10,

    NameAggrPair,

    'BLK00101C1A.UNIT2@OV23', OPH_TIMEAVERAGE,

    'BLK00101C1A.UNIT2@OV23', OPH_INTEGRATION,

    'BLK00101C1A.UNIT2@OV23', OPH_MINIMUM);
```


Complete

Complete - where complete processing options are provided for each named point including processing types, bit number and integration constants e.g. asking for 10 * 1 minute intervals with each point having a different processing type and parameters over a 10 minute period

Samples

Request four processing types for a packed point over a 10 minute period specifying IntervalCount of 2 so each interval is five minutes. Specify that Bit 3 of the packed point is to be used for the processing.

```
select * from processeddata (

    #06/04/2009 09:00:00#, #06/04/2009 10:00:00#, IntervalCount, 2,

    Complete,

    'BLK00101P1P.UNIT2@OV23', OPH_NOPBIT, OPH_GENERATESUMMARY, 1, 3,

    'BLK00101P1P.UNIT2@OV23', OPH_TIMEAVERAGE, OPH_GENERATESUMMARY, 1, 3,

    'BLK00101P1P.UNIT2@OV23', OPH_TIMESET, OPH_GENERATESUMMARY, 1, 3,

    'BLK00101P1P.UNIT2@OV23', OPH_TIMERESET, OPH_GENERATESUMMARY, 1, 3);
```

Request four processing types for an analog point over a 10 minute period specifying IntervalCount of 2 so each interval is five minutes. Specify integration constants of 1.0 and 3.5 to be used for the processing of the OPH_INTEGRATION processing.

```
select * from processeddata (

    #06/04/2009 09:00:00#, #06/04/2009 10:00:00#, IntervalCount, 2,

    Complete,

    'BLK00101C1A.UNIT2@OV23', OPH_TIMEAVERAGE, OPH_GENERATESUMMARY, 1, 1,

    'BLK00101C1A.UNIT2@OV23', OPH_MINIMUM, OPH_GENERATESUMMARY, 1, 1,

    'BLK00101R1A.UNIT2@OV23', OPH_INTEGRATION, OPH_GENERATESUMMARY, 1.0, 1,

    'BLK00101R1A.UNIT2@OV23', OPH_INTEGRATION, OPH_GENERATESUMMARY, 3.5, 1);
```

18.7.2 ProcDataColWise table

ProDataColWise UDF table provides functionality similar to the ProcessedData UDF table but columns are returned in a column wise orientation to facilitate use of data in report style ordering inside Reports or Excel.

TABLE NAME:- PROCDATACOLWISE	
COLUMN NAME	COLUMN DESCRIPTION
HANDLE	Point Handle.
ID	Points Unique ID.
TIMESTAMP	Timestamp in UTC .
TIME_NSEC	High resolution time stamp in nanoseconds.
NOP	Actual value of process point.
TIMEAVERAGE	Timed average over interval.
MAXIMUM	Maximum in interval.
MINIMUM	Minimum in interval.
MAXIMUMACTUALTIME_SECS	Time in seconds of maximum value in interval.
MAXIMUMACTUALTIME_NSECS	Time in nanoseconds of maximum value in interval.
MIMIMUMTIMEACTUAL_SECS	Time in seconds of minimum value of interval.
MINIMUMTIMEACTUALTIME_NSECS	Time in nanoseconds of minimum value of interval.
INTEGRATION	Integration over the interval.
TOGGLE	Count of digital transitions over interval.
TOGGLESET	Count of digital transitions to set state over interval.
TOGGLERESET	Count of digital transitions to reset state over interval.
TIMESET_SECS	Amount of time in seconds digital was set during interval.
TIMESET_NSECS	Amount of time in nsecs digital was set during interval.
TIMERESET_SECS	Amount of time in seconds digital was reset during interval.
TIMERESET_NSECS	Amount of time in nsecs digital was reset during interval.
TOTAL	Total of all samples during interval.
AVERAGE	Average of all samples during interval.
COUNT	Count all samples during interval.
STDEV	Std dev of all samples during interval.
END	Value at the end of the interval.
VARIANCE	Variance of all the samples during the interval.
RANGE	Total range of all values during the interval.

TABLE NAME:- PROCDATACOLWISE	
DURATIONGOOD_SECS	Amount of time secs in good quality during interval.
DURATIONGOOD_NSECS	Amount of time nsecs in good quality during interval.
DURATIONBAD_SECS	Amount of time secs in non-good quality during interval.
DURATIONBAD_NSECS	Amount of time nsecs in non-good quality during interval.
PERCENTGOOD	Percentage good quality during interval.
PERCENTBAD	Percentage non-good quality during interval.
DELTA	Difference between value at start and end of interval.
START	Value at the start of the interval.
STS	Status and Quality indicators for value fields

Samples

PointNames Only - Request all columns for three analog points over a 1 hour period specifying IntervalCount of 10 so each interval is six minutes.

```
select * from ProcDataColWise (

   #06/04/2009 09:00:00#, #06/04/2009 10:00:00#, IntervalCount, 10,

   PointNamesOnly,

   'BLK00101C1A.UNIT2@OV23', 'BLK00101C2A.UNIT2@OV23',
'BLK00101C3A.UNIT2@OV23');
```

Request selected columns for three analog points over a 1 hour period specifying IntervalCount of 10 so each interval is six minutes.

```
select ID, START, END, DELTA, AVERAGE, TIMEAVERAGE

 from ProcDataColWise (

   #06/04/2009 09:00:00#, #06/04/2009 10:00:00#, IntervalCount, 10,

   PointNamesOnly,

   'BLK00101C1A.UNIT2@OV23', 'BLK00101C2A.UNIT2@OV23',
'BLK00101C3A.UNIT2@OV23');
```

Request selected columns for three digital points over a 1 hour period specifying IntervalCount of 10 so each interval is six minutes.

```
select ID, START, END, TOGGLE, TOGGLESET, TOGGLERESET

 from ProcDataColWise (

   #06/04/2009 09:00:00#, #06/04/2009 10:00:00#, IntervalCount, 10,

   PointNamesOnly,

   'BLK00101M1D.UNIT2@OV23', 'BLK00101M2D.UNIT2@OV23',
'BLK00101M3D.UNIT2@OV23');
```

Complete - Request all columns for three analog points over a 1 hour period specifying IntervalCount of 10 so each interval is six minutes and specifying integration constants to be used.

```
select * from ProcDataColWise (

#06/04/2009 09:00:00#, #06/04/2009 10:00:00#, IntervalCount, 10,
```



```
Complete,

'BLK00101C1A.UNIT2@OV23',   OPH_GENERATESUMMARY, 1, 1,

'BLK00101R1A.UNIT2@OV23',   OPH_GENERATESUMMARY, 1.0, 1,

'BLK00101R1A.UNIT2@OV23', OPH_GENERATESUMMARY, 3.5, 1);
```

18.7.3 Summary tables

Whenever a request is made from the ProccessedData UDF table, or the ProcDataColWise UDF table the Summary UDF table is also populated and provides one set of summary data over the entire requested period of raw data. This table can be accessed via SQL requests but care must be taken to execute a ProcessedData or ProcDataColWise query to ensure the Summary table is properly populated.

TABLE NAME:- PROCESSEDDATASUMMARY	
COLUMN NAME	COLUMN DESCRIPTION
HANDLE	Point Handle
ID	Points Unique ID
COUNT	Number of intervals processed.
NUMSAMPLES	Number of raw samples processed.
STATUS	
ERROR	
AVERAGE	Average of all samples during entire requested time range.
MINIMUM	Minimum in entire requested time range.
MINTIME_SECS	Time in nanoseconds of minimum value in requested time range.
MINTIME_NSECS	Time in seconds of minimum value in requested time range.
MAXIMUM	Maximum in entire requested time range.
MAXTIME_SECS	Time in seconds of maximum value in requested time range.
MAXTIME_NSECS	Time in nanoseconds of maximum value in requested time range.
INTEGRATION	Integration over the in requested time range.
SUM	Sum of requested time range.
TOGGLE	Count of digital transitions over requested time range.
TOGGLESET	Count of digital transitions to set state over requested time range.
TOGGLERESET	Count of digital transitions to reset state over requested time range.
TIMESET_SECS	Amount of time in seconds digital was set over requested time range.
TIMESET_NSECS	Amount of time in nsecs digital was set over requested time range.

TIMERESET_SECS	Amount of time in seconds digital was reset over requested time range.
TIMERESET_NSECS	Amount of time in nsecs digital was reset over requested time range.

18.7.4 Scalar and Aggregate functions

Supported SQL functions

These supported Sql functions can help you query your historical data:

- Count (see page 300)
- Min (see page 301)
- Max (see page 301)
- Upper (see page 301)
- Lower (see page 301)
- Bitmask (see page 301)
- Bitextract (see page 301)
- Substring (see page 301)
- Len (see page 302)
- Concat (see page 302)
- Trim (see page 302)
- Ltrim (see page 302)
- Rtrim (see page 302)
- Abs (see page 302)
- Round (see page 302)
- Floor (see page 302)
- Ceiling (see page 302)
- Quality (see page 302)
- ISNULL (see page 302)
- NEW_TIME (see page 302)
- GETDATE (see page 302)

Note: *For more information on using the SQL functions to query your data, see the SQL_Funcs_Sample.ohs file that provides working examples of the various Scalar and Aggregate SQL functions supported by the OPH.*

Count

You can use the count function to return the exact number of database rows that match your name criteria:

Select count (*) from cfg_pt_name where name like 'BLK101%';

Min

You can use the min function to return the minimum values for a point during a time period that you set:

Select min(f_value) from pt_hf_hist where id=25 and timestamp> #07/01/2009 00:00:00# and timestamp< #07/01/2009 00:15:00#;

Max

You can use the max function to return the maximum value for a point during a time period that you set:

Select max(f_value) from pt_hf_hist where id=25 and timestamp> #07/01/2009 00:00:00# and timestamp< #07/01/2009 00:15:00#;

Upper

You can use the upper function to convert the capitalization of a string to all-upper-case.

Select name, id from cfg_pt_name where name=upper ('bLk00101c1a.UNIT2@OV23');

Lower

You can use the lower function to convert the capitalization of a string to all-lower-case.

Select lower (name), id from cfg_pt_name where name='BLK00101C1A.UNIT2@OV23';

Bitmask

You can use the bit mask function to select bits that match your criteria, and optionally shift the result to the right (in this case, the quality bits 8 and 9 are shifted to a single-digit number by using '8'):

Select id, timestamp, f_value, sts, BITMASK (sts, 0x0300, 8) from pt_hf_hist where id=25 and timestamp>#07/01/2009 00:00:00# and timestamp< #07/01/2009 00:15:00#;

Bitextract

You can use the bit extract function to extract the value of a specific bit (which, in this example, means samples where limit checking is off):

Select id, timestamp, f_value, sts from pt_hf_hist where id=25 and timestamp> #07/01/2009 00:00:00# and timestamp< #07/01/2009 00:15:00# and BITEXTRACT (sts, 12) =1;

Substring

Substring clips out a piece of a string beginning at a start byte and going for count bytes

select name, id, data_type from cfg_pt_name where substring(name,0,9) = 'BLK00101C';

Len

String length.

select name, id, len(name) as len from cfg_pt_name where len(name) > 25;

Concat

Return concatenation of two strings

Trim

Remove leading and trailing spaces.

LTrim

Left Trim trims all the occurrences of any one of a set of characters off of the left side of a string.

RTrim

Right Trim trims all the occurrences of any one of a set of characters off of the right side of a string.

Abs

Absolute value of a floating point column.

Round

Round of a floating point column.

Floor

Floor is the largest integer smaller than or equal to value.

Ceiling

Ceil is the smallest integer higher than or equal to value.

Quality

Indicates point condition Good, Fair, Poor, Bad, or Timed Out.

ISNULL

IS NULL tests column (or an expression) for the absence of any data.

NEW_TIME

Convert the time and date from one time zone to another time zone.

GETDATE

Gets the current date and time.

18.8 History Console tool syntax

Use the correct syntax in order to get the most information from the history console tool. The following sections include support, server, retrieval, archive, and lab commands that might be helpful to you. There is also a section on additional syntax terms.

18.8.1 Support commands

There are three support commands: **set, echo** and **run.**

Set

Use the **set** command to define a runtime macro:

Set $<Macro>=<Value >;

PARAMETER NAMES	DESCRIPTIONS
Macro	Name of the macro (required)
Value	Value to replace the macro instance (required)

Example of the set command:

```
Set $SERVER=PHPWS-01; //macro definition

Connect Server=$SERVER; //instance of $SERVER is replaced by PHPWS-01 during runtime;
```

A macro gets replaced by its defined value during runtime. A macro can only be used for command parameter values. A macro is valid until another set command is encountered with the same macro name. Macros are recursive in that they apply to calls inside a script file as well. Macros should be included in the runtime selection to take effect for commands that follow.

Echo

Use the **echo** command to echo specified text to the output stream:

Echo Text=<text>;

PARAMETER NAME	DESCRIPTION
Text	The text that you want to echo to the output stream (required).

Example of the echo command:

```
Echo Text=Hello World!;//displays Hello World! to the output
```

The echo command is useful for tagging the output stream with text messages. For example, you might use it to describe the expected result of a subsequent query.

Run

Use the **run** command to load a script file and runs its contents (and optionally dump the results to another file):

Run Source=<source file>, [Results=<results file>];

PARAMETER NAMES	DESCRIPTIONS
Source	Complete path and filename to the source script file that you want to run (required).
Results	Complete path and filename of the file that you want the results written to (optional).

Use the run command to execute previously saved Console scripts. By default, the results of the execution are shown in the result pane but you can specify an optional Results file to dump the results to.

18.8.2 Server commands

There are two server commands: **connect** and **disconnect.**

Connect

Use the **connect** command to open a connection to a Historian server:

Connect Server=<server name>, [User=<user name>], [Password=<password>];

PARAMETER NAMES	DESCRIPTIONS
Server	The Historian server that you want to connect to (required)
User	Login name, which defaults to an empty string (optional)
Password	Password, which defaults to an empty string (optional)

After a connection has been established, the connection will remain available for all commands until a disconnect command is called. Another call to connect while a previous connection is established will disconnect the previous connection and create a new one based on the new call.

Config

This command configures the server retrieval mode. Use this command to establish the retrieval type. Once a mode has been established, the mode type will remain the same for all commands until a second configuration command is encountered or the connect object is deleted.

Syntax:

Config Mode= <mode type> ;

PARAMETER NAME	DESCRIPTION
Mode	(Optional) The requested mode type. The default is "MODE_LATEST"

These are the choices for this command:

- MODE_LATEST
- MODE_ORIGINAL
- MODE_ALL
- MODE_MODIFIED

Disconnect

Use the **disconnect** command to close a connection to a Historian server:

Disconnect;

An error is raised if no previous connection is established and a call to disconnect is made.

18.8.3 Retrieval commands

There are nine retrieval commands:

- SQL (see page 305)
- GetPointConfig (see page 305)
- GetPointAttributes (see page 306)
- SyncReadRaw (see page 307)
- SyncReadProcessed (see page 308)
- SyncReadRealTime (see page 309)
- ReadAlarms (see page 310)
- ReadTextMsgs (see page 311)
- ReadSoe (see page 312)

Sql

Use the **sql** command to execute a Sql command:

Sql command=<Sql statement>;

PARAMETER NAMES	DESCRIPTIONS
Sql statement	The Sql statement that you want to execute (required)

Example of the sql command:

```
Sql command=Select * From CFG_PT_NAME;
```

GetPointConfig

Use the **GetPointConfig** command to retrieve point configuration data from the historian:

GetPointConfig Points={<list of points>};

PARAMETER NAMES	DESCRIPTIONS
Points	The list of point names, separated by commas. The default is all

PARAMETER NAMES	DESCRIPTIONS
	points if you do not specify specific points (optional).

Use the GetPointConfig command to retrieve the configuration information for each point specified in the points parameter. If no points are specified, configuration data for ALL points in the selected historian are returned.

GetPointAttributes

Use the **GetPointAttributes** command to retrieve point attribute data from the historian:

GetPointAttributes

Points={<list of points>},

Range={<start time>, <end time>};

PARAMETER NAMES	DESCRIPTIONS
Points	The list of point names, separated by commas (required)
Range	Specifies the start time and end time in **mm/dd/yyyy hh:mm:ss** format (optional). If you do not specify these times, end time is set to NULL, which returns the current attributes for the specified points.

Example of the GetPointAttributes command:

```
GetPointAttributes

      Points={Point1.Unit1@Net1, Point2.Unit1@Net1},

      Range={06/01/2006 10:00:00, 06/01/2006 11:00:00};
```

This example will retrieve all attribute values for Point1 and Point2 within 10am to 11am of June 1, 2006.

Use the GetPointAttributes command to retrieve point attribute information for each point specified in the points parameter. If current attributes are desired, set end time to NULL or alternatively, do not specify the range parameter. If latest attributes relative to a given time are desired, set start time to NULL and end time to the reference time.

SyncReadRaw

Use the **SyncReadRaw** command to retrieve raw (unprocessed) data synchronously from the historian:

SyncReadRaw

Range={<start time>, <end time>},

Points={<list of points>},

[Filters={<list of filters>}],

[Bound=<true/false>],

[Options={<list of options>}],

[Bits={<list of bit positions>}],

[Constants={<list of constants>}];

PARAMETER NAMES	DESCRIPTIONS
Range	Specifies the start time and end time in **mm/dd/yyyy hh:mm:ss** format (required).
Points	The list of point names, separated by commas (required).
Filters	List of "or" filters per point specified in the Points parameter, separated by commas (optional). Defaults to RC_RAW_DATA_VAL if not specified. Refer to the list of filters (see page 316) for details.
Bound	Specifies whether bounding values are returned or not (optional). Defaults to false.
Options	List of "or" options per point specified in the Points parameter, separated by commas (optional). Defaults to "no options" if not specified, which in this case means no summary and no glitch detection. Refer to the list of options (see page 316) for details.
Bits	Comma-separated list of bit positions per point specified in the Points parameter (optional). Defaults to zero.
Constants	Comma-separated list of constants per point specified in the Points parameter (optional). Defaults to one.

Example of the SyncReadRaw command:

```
SyncReadRaw

        Range={06/01/2006 10:00:00, 06/01/2006 11:00:00},

        Points={Point1.Unit1@Net1, Point2.Unit1@Net1};
```

This example will retrieve the raw values for Point1 and Point2 within 10am to 11am of June 1, 2006 with default options.

Use the SyncReadRaw command to synchronously retrieve raw values for a given set of points within a given time domain. Note that the entries for Filters, Options, Bits and Constants must correspond to an entry in the list of Points. After you specify one of the optional lists, it must contain an entry for every corresponding entry on the list of points (a one-to-one relationship). To make an equivalent asynchronous call, call the AsyncReadRaw command.

SyncReadProcessed

Use the **SyncReadProcessed** command to retrieve processed data synchronously from the historian:

SyncReadProcessed

Range={<start time>, <end time>},

Interval=<interval>,

Points={<list of points>},

[Aggregates={<list of aggregates>},]

[Options={<list of options>}],

[Bits={<list of bit positions>}],

[Constants={<list of constants>}];

PARAMETER NAMES	DESCRIPTIONS
Range	Specifies the start time and end time in **mm/dd/yyyy hh:mm:ss** format (required).
Interval	Size or number of the intervals (required). You can specify the interval either as a time value (hh:mm:ss.uu), as a constant (refer to the list of interval constants (see page 316)), or as a whole number value.
Points	The list of point names, separated by commas (required).
Aggregates	List of aggregate operations to be applied for each entry in the Points parameter (optional). Defaults to OPH_NOP if not specified. Refer to the list of aggregates. (see page 316)
Options	List of "or" options per point specified in the Points parameter, separated by commas (optional). Defaults to "no options" if not specified, which in this case means no summary and no glitch detection. Refer to the list of options. (see page 316)

PARAMETER NAMES	DESCRIPTIONS
Bits	Comma-separated list of bit positions per point specified in the Points parameter (optional). Defaults to zero.
Constants	Comma-separated list of constants per point specified in the Points parameter (optional). Defaults to one.

Example of the SyncReadProcessed command:

```
SyncReadProcessed

        Range={06/01/2006 10:00:00, 06/01/2006 11:00:00},

        Interval=60,

        Points={Point1.Unit1@Net1, Point2.Unit1@Net1},

        Aggregates={OPH_TIMEAVERAGE, OPH_INTEGRATION};
```

This example will retrieve the average and integration for Point1 and Point2, respectively, per minute within 10am to 11am of June 1, 2006 with default options.

Use this command to synchronously retrieve processed and periodic values for a given set of points within a given time domain. Note that the entries for Aggregates, Options, Bits and Constants must correspond to an entry in the list of Points.

SyncReadRealTime

This command retrieves near real-time data synchronously from the Historian for a given set of points.

Syntax:

SyncReadRealTime

Points={<list points>};

PARAMETER NAME	DESCRIPTION
Points	(Required) Comma-separated list of point names

Example:

SyncReadRealTime

Points={Point1.Unit1@Net1, Point2.Unit1@Net1};

This example will retrieve the raw values for Point1 and Point2 in the nearest real time values.

ReadAlarms

Use the **ReadAlarms** command to retrieve alarms from the historian:

ReadAlarms

Range={<start time>, <end time>},

[Point=<point name substring>],

[Source=<source id>],

[System=<system id>],

[Drop=<drop id>],

[RecordType=<record type>],

[AlarmType=<alarm type>];

PARAMETER NAMES	DESCRIPTIONS
Range	Specifies the start time and end time in **mm/dd/yyy hh:nn:ss** format (required).
Point	Point name substring (optional). Accepts wild cards. Defaults to an empty string or no point filter.
Source	Source ID (optional). Defaults to no source ID filter.
System	System ID (optional). Defaults to no system ID filter.
Drop	Drop ID (optional). Defaults to no drop filter.
RecordType	Type of record (optional). Refer to the list of record types (see page 316). Defaults to no record type filter.
AlarmType	Type of alarm (optional). Refer to the list of alarm types (see page 316). Defaults to no alarm type filter.

ReadTextMsgs

Use the **ReadTextMsgs** command to retrieve text messages from the historian:

ReadTextMsgs

Range={<start time>, <end time>},

[Point=<point name substring>],

[Source=<source id>],

[System=<system id>],

[Drop=<drop id>],

[MsgType=<message type>],

[EventType={<list of event types>}];

PARAMETER NAMES	DESCRIPTIONS
Range	Specifies the start time and end time in **mm/dd/yyy hh:nn:ss** format (required).
Point	Point name substring (optional). Accepts wild cards. Defaults to an empty string or no point filter.
Source	Source ID (optional). Defaults to no source ID filter.
System	System ID (optional). Defaults to no system ID filter.
Drop	Drop ID (optional). Defaults to no drop filter.
MsgType	The string value that specifies the type of message that you want to retrieve--'SYSTEM' or 'OPEVENT (optional). Defaults to no message type filter.
EventTypes	List of event types, separated by commas (optional). Refer to the syntax terms (see page 316) for details. Defaults to a no events type filter.

Example of the ReadTextMsgs command:

```
ReadTextMsgs

      Range={06/01/2006 10:00:00, 06/01/2006 11:00:00},

      MsgType='OPEVENT';
```

This example will retrieve operator event messages between 10am to 11am of June 1, 2006.

Use the MsgType parameter to specify the type of message that you want to retrieve. For example, to retrieve operator events, set MsgType to 'OPEVENT'. To retrieve system messages, set MsgType to 'SYSTEM'. If the MsgType parameter is not supplied or is set to a blank string, all text messages are returned.

ReadSoe

Use the **ReadSoe** command to retrieve sequence of events from the historian:

ReadSoe

Range={<start time>, <end time>},

[Point=<point name substring>],

[Source=<source id>],

[System=<system id>],

[Drop=<drop id>],

PARAMETER NAMES	DESCRIPTIONS
Range	Specifies the start time and end time in **mm/dd/yyy hh:nn:ss** format (required).
Point	Point name substring (optional). Accepts wild cards. Defaults to an empty string or no point filter.
Source	Source ID (optional). Defaults to no source ID filter.
System	System ID (optional). Defaults to no system ID filter.
Drop	Drop ID (optional). Defaults to no drop filter.

18.8.4 Archive commands

There are two archive commands:

- GetDeviceInfo
- CommandDevice

GetDeviceInfo

Use the **GetDeviceInfo** command to retrieve the list of available archive devices from the historian:

GetDeviceInfo;

CommandDevice

Use the **CommandDevice** command to initiate an archive device command:

CommandDevice Device=<device id>, Action=<action>;

PARAMETER NAMES	DESCRIPTIONS
Device	The ID of the device that you want to send the command to (required).
Action	The action that you want to execute (required). Refer to the list of

PARAMETER NAMES	DESCRIPTIONS
	commands. (see page 316)

18.8.5 Lab command

There is one lab command:

- SendLabFile

SendLabFile

Use the **SendLabFile** command to send a lab file to the historian:

SendLabFile File=<source lab file>, [Results=<results file>];

PARAMETER NAMES	DESCRIPTIONS
File	The complete path and file name of the lab file that you want to send (required).
Results	The complete path and file name of the file that you want the results written to (optional).

By default, the results of the processing per row are shown, but you can specify an optional Results file to dump the results to.

18.8.6 File history commands

There are 4 file history commands:

- CreateDir
- PutFile
- GetFile
- GetList

CreateDir

Use this command to create a directory in the historian server.

Syntax:

CreateDir Pathname=<full directory name>, Timestamp=<timestamp>;

PARAMETER NAMES	DESCRIPTIONS
Pathname	(Required) Complete pathname of directory to be created
Timestamp	(Required) Specifies the time stamp in **mm/dd/yyyy hh:mm:ss.nnnnnnnnn** format.

PutFile

Use this command to put a file in the historian server.

Syntax:

Putfile Source=<source filename>,

Destination=<destination filename>,

Timestamp=<timestamp>;

PARAMETER NAMES	DESCRIPTIONS
Source	(Required) Complete pathname of the source file to be sent
Destination	(Required) Complete pathname of the destination file located in the server
Timestamp	(Required) Specifies the time stamp in **mm/dd/yyyy hh:mm:ss.nnnnnnnnn** format

GetFile

Use this command to retrieve files from the historian server.

Syntax:

Getfile Source=<source filename or sequence number>, Destination=<destination filename>, Timestamp=<timestamp>;

PARAMETER NAMES	DESCRIPTIONS
Source	(Required) Complete pathname or sequence number of the source file to be retrieved
Destination	(Required) Complete pathname of the destination file
Timestamp	(Required) Specifies the time stamp in **mm/dd/yyyy hh:mm:ss.nnnnnnnnn** format

GetList

Use this command to retrieve directory information from the historian server.

Syntax:

Getlist Range= {<start time>, <end time>},

Pathname=<full directory name>,

[Domain =<domain name>],

[Computer =<computer name>],

[Application =<application name>],

[User =<user name>];

PARAMETER NAMES	DESCRIPTIONS
Range	(Required) Specifies the time range (start and end time) in **mm/dd/yyyy hh:nn:ss.nnnnnnnnn** format
Pathname	(Required) Complete pathname of the directory to be created
Domain	(Optional) Domain name
Computer	(Optional) Computer name
Application	(Optional) Application name
User	(Optional) User name

18.8.7 Syntax terms

SyncReadRaw filters:

- RC_VALUE_LIMITS
- RC_SCAN_REMOVED
- RC_ENTERED_VALUE
- RC_TIMED_OUT
- RC_BAD_QUALITY
- RC_FAIR_QUALITY
- RC_RAW_DATA_VAL
- RC_INITIAL_VAL
- RC_EXTERNAL_CALIBRATION
- RC_CHECK_REMOVED
- RC_SENSOR_ALARM
- RC_IN_ALARM
- RC_POOR_QUALITY
- RC_GOOD_QUALITY
- RC_CUTOUT_ALM
- RC_RETURN_VAL

SyncReadProcessed aggregates:

- OPH_NOP — F.VALUE this is numeric
- OPH_NOPBIT — F.VALUE
- OPH_TIMEAVERAGE F.VALUE
- OPH_MAXIMUM F.VALUE
- OPH_TOGGLE - TIME
- OPH_TIMESET TIME
- OPH_TOGGLERESET — TIME this is string
- OPH_MINIMUM F.VALUE
- OPH_MAXIMUMACTUALTIME TIME
- OPH_MINIMUMACTUALTIME TIME
- OPH_INTEGRATION
- OPH_TOGGLESET TIME
- OPH_TIMERESET TIME
- OPH_TOTAL
- OPH_AVERAGE F.VALUE
- OPH_COUNT
- OPH_START
- OPH_END
- OPH_DELTA
- OPH_RANGE

- OPH_PERCENTBAD
- OPH_PERCENTGOOD
- OPH_DURATIONBAD
- OPH_DURATIONGOOD
- OPH_STDEV
- OPH_VARIANCE
- OPH_REGDEV
- OPH_REGSLOPE
- OPH_REGCONST
- OPH_ANNOTATION

SyncReadRaw / SyncReadProcessed options:

- OPH_GENERATESUMMARY
- OPH_DETECTGLITCH

ReadAlarms alarm types:

- ALMTYPE_SENSOR
- ALMTYPE_HIGH
- ALMTYPE_HI_WRS
- ALMTYPE_LOW
- ALMTYPE_LOW_BET
- ALMTYPE_LOW_WRS_UDA
- ALMTYPE_SP_ALM
- ALMTYPE_SID_ALM
- ALMTYPE_ST_CHG
- ALMTYPE_HI_BET
- ALMTYPE_HI_UDA
- ALMTYPE_HI_WRS_UDA
- ALMTYPE_HI_BET_UDA
- ALMTYPE_LOW_WRS
- ALMTYPE_LOW_UDA
- ALMTYPE_LOW_BET_UDA
- ALMTYPE_TIME_OUT
- ALMTYPE_ALARM
- ALMTYPE_INCR_ALARM

ReadTxtMsgs event types:

- OPEVENT
- SYSTEM