Wonderware Historian Database Reference

Invensys Systems, Inc.

Revision F
Last Revision: December 3, 2009

Copyright

Invensys Systems, Inc.
26561 Rancho Parkway South
Lake Forest, CA 92630 U.S.A.
(949) 727-3200

http://www.wonderware.com

For comments or suggestions about the product documentation, send an e-mail message to productdocs@wonderware.com.

Trademarks

Contents

Welcome 13
Wonderware Historian Documentation Set 13
Documentation Conventions 14
Technical Support 15

Chapter 1 Table Categories 17
History Tables 17
History Table Format 19
"Wide" History Table Format 20
"Live" Table Format 22
Event Tables 22
InTouch Integration Tables 23
Modification Tracking Tables 23
Namespaces and Grouping Tables 23
Tag Definition Tables 24
Replication Tables 24
System Configuration Tables 25
ArchestrA Browsing Tables 25

Chapter 2 Tables 27
Accessing Information About a Database Table 27
aaAreaData 28
aaAreaXML 28

aaAttributeData 29
aaAttributeDataPending 29
aaObjectData 29
aaObjectDataPending 30
ActionType 31
AnalogSnapshot 31
AnalogSummaryHistory (INSQL.Runtime.dbo.AnalogSummaryHistory) 32
AnalogSummaryTag 36
AnalogTag 37
Annotation 39
AttributeType 40
CalcType 40
ConfigStatusPending 40
ConfigStatusSnapshot 41
Context 41
CurrentEditor 42
CustomReplicationSchedule 42
DetectorType 43
Deviation 44
DiscreteSnapshot 45
DiscreteTag 45
EngineeringUnit 46
ErrorLog 47
EventHistory 48
EventTag 49
EventTagPendingDelete 51
Frequency 51
History (INSQL.Runtime.dbo.History) 52
HistoryBlock (INSQL.Runtime.dbo.HistoryBlock) 59
HistorianSysObjects 61
IntervalReplicationSchedule 61
InTouchNode 62
InTouchSpecific 63
IODriver 64
IOServer 68
IOServerType 70
Limit 70
LimitName 71
Live (INSQL.Runtime.dbo.Live) 72

LocalizedText 74
Message 74
ModLogColumn 75
ModLogTable 75
OPCQualityMap 76
PrivateGroupTag 77
PrivateNameSpace 77
PublicGroupTag 78
PublicNameSpace 78
QualityMap 79
RateOfChange 80
ReplicationGroup 80
ReplicationSchedule 82
ReplicationScheduleType 82
ReplicationServer 83
ReplicationSyncRequest 85
ReplicationTag 86
ReplicationTagEntity 86
ReplicationType 89
ServerList 89
SnapshotDetail 89
SnapshotTag 90
SQLTemplate 91
StateSummaryHistory (INSQL.Runtime.dbo.StateSummaryHistory) 91
StateWideHistory (INSQL.Runtime.dbo.StateWideHistory) 95
StorageLocation 99
StorageNode 101
StringSnapshot 102
StringTag 103
StructureAttributes 103
StructureTag 104
StructureType 104
SummaryData 104
SummaryHistory 105
SummaryOperation 107
SummaryTagList 108
SystemParameter 108
Tag 109
TagRef 113

TagType 114
TimeDetectorDetail 114
TimeDetectorDetailPendingDelete 115
TimeZone 115
Topic 116
TopicImportInfo 117
UserDetail 118
WideHistory (INSQL.Runtime.dbo.WideHistory) 119

Chapter 3 Views 125
History Table Views 125
v_EventSnapshot 126
v_EventStringSnapshot 127
v_ModTracking 128
v_SnapshotData 129
ReplicationSyncRequestInfo 130
Millisecond Resolution Differences in SQL Server 2008 135

Chapter 4 Stored Procedures 137
Stored Procedures 137
aaActionStringSelect 138
aaAddAnalogSummaryTag 138
aaAddReplicationGroup 142
aaAddReplicationSchedule 143
aaAddReplicationServer 144
aaAddReplicationTagEntity 145
aaAddStateSummaryTag 146
aaAddStructureTag 149
aaAddTag 152
aaAnalogDetail 154
aaAnalogTagDelete 154
aaAnalogTagInsert 154
aaAnalogTagSelect 159
aaAnalogTagUpdate 159
aaAnnotationDelete 160
aaAnnotationInsert 160
aaAnnotationRetrieve 161
aaAnnotationSelect 162
aaAnnotationUpdate 162
aaArchestrANSClear 162

aaCleanupAfterCommit 163
aaCleanupSystemNotRunning 163
aaCommitChanges 164
aaCommitChangesAtStartup 164
aaContextDelete 164
aaContextInsert 165
aaContextSelect 165
aaContextUpdate 165
CreateReplicationServerDefaultGroups 166
CreateReplicationServerSystemTags 166
aaDBChangesPending 166
aaDBConfig 167
aaDeleteOlderEvents 167
aaDeleteOlderSummaries 167
aaDeleteReplicationGroup 168
aaDeleteReplicationSchedule 168
aaDeleteReplicationServer 169
DeleteReplicationServerSystemTags 169
aaDeleteReplicationTagEntity 169
aaDeleteTag 170
aaDetectorStringSelect 170
aaDiscreteDetail 170
aaDiscreteTagDelete 171
aaDiscreteTagInsert 171
aaDiscreteTagSelect 175
aaDiscreteTagUpdate 175
aaEngineeringUnitDelete 176
aaEngineeringUnitInsert 176
aaEngineeringUnitSelect 177
aaEngineeringUnitUpdate 178
aaEventDetection 178
aaEventHistoryInsert 179
aaEventHistorySelect 180
aaEventSnapshotInsert 181
aaEventSnapshotSelect 181
aaEventTagDelete 182
aaEventTagDetail 182
aaEventTagInsert 183
aaEventTagSelect 186
aaEventTagSelectAll 186
aaEventTagSelectDeleted 186
aaEventTagSelectDisabled 186
aaEventTagSelectInserted 187

aaEventTagSelectUpdated 187
aaEventTagUpdate 187
aaGetAnalogSummaryTags 188
aaGetDbRevision 188
aaGetLastTagKey 188
aaGetReplicationGroups 189
aaGetReplicationNamingParameters 190
aaGetReplicationSchedules 190
aaGetReplicationServers 191
aaGetReplicationTagEntities 192
aaGetReplicationTags 193
aaGetStateSummaryTags 193
aaHistorianConfigNSExpand 194
aaHistorianNSExpand 194
aaHistorianStatusSelect 194
aaHistorianStatusSet 195
aaHistoryBlockSelect 196
aaInTouchNodeTagList 196
aaIODriverDelete 197
aaIODriverInsert 197
aaIODriverSelect 200
aaIODriverUpdate 200
aaIOServerDelete 201
aaIOServerInsert 201
aaIOServerSelect 203
aaIOServerTypeDelete 203
aaIOServerTypeInsert 203
aaIOServerTypeSelect 204
aaIOServerTypeUpdate 204
aaIOServerUpdate 205
aaLimitDelete 205
aaLimitInsert 205
aaLimitNameDelete 206
aaLimitNameInsert 207
aaLimitNameSelect 207
aaLimitNameUpdate 207
aaLimitSelect 208
aaLimitUpdate 208
aaMessageDelete 208
aaMessageInsert 209
aaMessageSelect 209
aaMessageUpdate 209
aaModLogStatus 210

aaPrivateNSAddGroup 210
aaPrivateNSAddLeaf 211
aaPrivateNSDeleteGroup 211
aaPrivateNSDeleteLeaf 212
aaPrivateNSExpand 212
aaPrivateNSSelect 213
aaPrivateNSUpdateGroup 213
aaPublicNSAddGroup 213
aaPublicNSAddLeaf 214
aaPublicNSDeleteGroup 215
aaPublicNSDeleteLeaf 215
aaPublicNSExpand 216
aaPublicNSSelect 216
aaPublicNSUpdateGroup 216
aaRedirectToInTouch 217
aaSetAISamples 218
aaSetCalculatedAISamples 218
aaSetServerTimeStamp 219
aaSetStorageRule 219
aaSetTagStorage 221
aaSnapshotDetailSelect 222
aaSnapshotDetailUpdate 223
aaSnapToSummary 223
aaSpaceManager 224
aaStorageLocationSelect 224
aaStorageLocationUpdate 225
aaStringDetail 226
aaStringTagDelete 226
aaStringTagInsert 227
aaStringTagSelect 230
aaStringTagUpdate 230
aaSummaryActionInsert 231
aaSummaryDetail 231
aaSummaryOperationDelete 232
aaSummaryOperationInsert 232
aaSummaryOperationSelect 233
aaSummaryOperationUpdate 234
aaSummaryTagListDelete 234
aaSummaryTagListInsert 235
aaSummaryTagListSelect 236
aaSummaryTagListUpdate 236
aaSystemConfigNSExpand 236
aaSystemNSExpand 237

aaSystemNSExpand2 238
aaSystemParameterSelect 238
aaSystemParameterUpdate 238
aaTagConfig 239
aaTagConfigModified 239
aaTagConfigSelect 240
aaTagInfo 240
aaTagType 241
aaTimeDetectorDetailInsert 241
aaTimeDetectorDetailSelect 242
aaTimeDetectorDetailUpdate 242
aaTopicDelete 243
aaTopicInsert 243
aaTopicSelect 244
aaTopicUpdate 244
aaUpdateCalculatedAISamples 245
aaUserAccessLevelSelect 246
aaUserDetailUpdate 246
Extended Stored Procedures 246
History Extended Stored Procedures 247
xp_AnalogHistory 247
xp_AnalogHistoryDelta 247
xp_AnalogWideHistory 248
xp_AnalogWideHistoryDelta 248
xp_DiscreteHistory 248
xp_DiscreteHistoryDelta 248
xp_DiscreteWideHistory 248
xp_DiscreteWideHistoryDelta 249
Utility Extended Stored Procedures 249
xp_DiskCopy 249
xp_NewHistoryBlock 250
xp_ProcList 250
xp_RescanHistoryBlocks 250
xp_SetStorageTimeDeadband 250
xp_SetStorageValueDeadband 251
xp_SetStoreForwardEvent 251
Extended Stored Procedure Arguments 252
Literal Date Expressions 253
GetDate() Expressions 254
DateAdd(...) Expressions 254
System Extended Stored Procedures 255
xp_AllowCommit 255
xp_TZgetdate 256

Stored Procedures for Internal Use 256
Creating Stored Procedures 258

Chapter 5 User-Defined Functions 261
faaCheckLicenseViolation 261
faaContainedName 262
faaGetHistorianTagNames 262
faaGetHierarchicalAttributeNames 262
faaGetLocalizedText 263
faaLicensedTagDetails 263
faaLicensedTagTotal 264
faaObjectTagName 264
faaTagsInLicenseViolation 265
faaTZgetdate 266
faaUser_ID 266

Chapter 6 Backward Compatibility Entities 269
Backward Compatibility Views 269
History Table Views 269
Summary Views 270
v_SummaryData 271
NamedSystemParameter 272
SystemNameSpace 273
InSQLSysObjects 274
v_ErrorLog 274
Backward Compatibility Tables 275
AnalogHistory (INSQL.Runtime.dbo.AnalogHistory) 276
AnalogLive (INSQL.Runtime.dbo.AnalogLive) 277
AnalogWideHistory 277
ComplexHistory 278
ComplexTag 279
DiscreteHistory (INSQL.Runtime.dbo. DiscreteHistory) 280
DiscreteLive (INSQL.Runtime.dbo.DiscreteLive) 280
DiscreteWideHistory 281
GroupTagList 282
ManualAnalogHistory 282
ManualDiscreteHistory 283
ManualStringHistory 283
NameSpaceIcons 284
StringHistory (INSQL.Runtime.dbo.StringHistory) .. 285

StringLive (INSQL.Runtime.dbo.StringLive) 286
StringWideHistory 286
TagGroup 287
WideTableDictionary 287
Renamed Tables 288
Backward Compatibility Stored Procedures 288
aaAnalogDetail 289
aaDiscreteDetail 289
aaStringDetail 290
ww_CheckClientVersion 290
ww_CheckWhichDb 291
ww_dbCheck 291
ww_DBConfig 291
ww_LoadInSQLProcedureBody 292
ww_MDASAnalogTagInsert 292
ww_MDASAnalogTagUpdate 292
ww_MDASDiscreteTagInsert 292
ww_MDASDiscreteTagUpdate 292
ww_MDASStringTagInsert 293
ww_MDASStringTagUpdate 293
Renamed Stored Procedures 293
Backward Compatibility Functions 302

Index 303

Welcome

This guide describes the database model of the Wonderware Historian system. Each database entity is described, and the relationships between the entities are defined. It is very important that you understand these data structures and relationships to effectively query Wonderware Historian and build productive client applications that interact with it.

Wonderware Historian Documentation Set

The Wonderware Historian documentation set includes the following guides:

- *Wonderware Historian Installation Guide* (InSQLInstall.pdf). This guide provides information on installing the Wonderware Historian, including hardware and software requirements and migration instructions.
- *Wonderware Historian Concepts Guide* (InSQLConcepts.pdf). This guide provides an overview of the entire Wonderware Historian system and describes each of the subsystems in detail.
- *Wonderware Historian Administration Guide* (InSQLAdmin.pdf). This guide describes how to administer and maintain an installed Wonderware Historian, such as configuring data acquisition and storage, managing security, and monitoring the system.
- *Wonderware Historian Database Reference* (InSQLDatabase.pdf). This guide provides documentation for all of the Wonderware Historian database entities, such as tables, views, and stored procedures.

- *Wonderware Historian Glossary* (InSQLGlossary.pdf). This guide provides definitions for terms used throughout the documentation set.

In addition, the *Wonderware® ArchestrA License Manager Guide* (License.pdf) describes the ArchestrA License Manager and how to use it to install, maintain, and delete licenses and license servers on local and remote computers.

A PDF file for each of these guides is available on the Wonderware Historian installation CD. You can easily print information from the PDF files. The Wonderware Historian documentation is also provided as an online help file, which can be accessed from the System Management Console management tool.

Documentation Conventions

This documentation uses the following conventions:

Convention	Used for
Initial Capitals	Paths and file names.
Bold	Menus, commands, dialog box names, and dialog box options.
`Monospace`	Code samples and display text.

Technical Support

Wonderware Technical Support offers a variety of support options to answer any questions on Wonderware products and their implementation.

Before you contact Technical Support, refer to the relevant section(s) in this documentation for a possible solution to the problem. If you need to contact technical support for help, have the following information ready:

- The type and version of the operating system you are using.
- Details of how to recreate the problem.
- The exact wording of the error messages you saw.
- Any relevant output listing from the Log Viewer or any other diagnostic applications.
- Details of what you did to try to solve the problem(s) and your results.
- If known, the Wonderware Technical Support case number assigned to your problem, if this is an ongoing problem.

Chapter 1

Table Categories

There are eight table categories within the Wonderware Historian Runtime database. Tables in a category together facilitate a particular functionality in the historian.

Note Additional tables and views are provided for backward compatibility support. For more information, see Chapter 6, "Backward Compatibility Entities."

History Tables

Because normal Microsoft SQL Server functionality cannot handle the storage and retrieval of huge quantities of rapidly changing data, plant data storage and retrieval are made possible by the Wonderware Historian storage subsystem, the history tables, and the retrieval system.

Some of the history tables are implemented as normal SQL Server tables, and the information contained in them is stored in the Runtime database file (Run100Dat.mdf). Others are implemented as a special type of table called a remote table, or extension table. Extension tables do not actually exist in the database, but rather expose data that is stored in special history files (history blocks) on disk using OLE DB technology.

For more information, see Chapter 6, "Data Retrieval Subsystem," in your *Wonderware Historian Concepts Guide*.

Acquired tag data can be presented in the history tables in four different formats: a normal historical format, a "wide" format, a "live" format, and an analog/state summary history format. Information about the history blocks is stored in the special HistoryBlock extension table.

To make querying from the history tables easier, views have been provided. Instead of specifying the table name using the required four-part syntax (INSQL.Runtime.dbo.<tablename>), you can simply use the view name instead. The history tables and associated views are listed in the following table. (Backward compatibility tables and views are not included.)

Table Name (OLE DB Provider Syntax)	Associated View
History (INSQL.Runtime.dbo.History)	History
WideHistory (INSQL.Runtime.dbo.WideHistory)	(none)
StateWideHistory (INSQL.Runtime.dbo.StateWideHistory)	(none)
Live (INSQL.Runtime.dbo.Live)	Live
HistoryBlock (INSQL.Runtime.dbo.HistoryBlock)	HistoryBlock
AnalogSummaryHistory (INSQL.Runtime.dbo.AnalogSummaryHistory)	AnalogSummaryHistory
StateSummaryHistory (INSQL.Runtime.dbo.StateSummaryHistory)	StateSummaryHistory

The History and Live tables can accommodate a mixture of tag types and should be used for all queries. The vValue column returns a sql_variant for all tag types. The Value column returns a float value for analog and discrete tags and a NULL for string tags. The Value column is included to allow for aggregation and other operations that are not permitted on a sql_variant column.

You can relate these tables to other tables in the Wonderware Historian database.

For more information on each of these tables, see the corresponding table description in this documentation.

Note The AnalogHistory, DiscreteHistory, StringHistory, AnalogLive, DiscreteLive, and StringLive tables are provided for backward compatibility and can only accept tagnames in the SELECT statement that are of the same type; that is, you cannot mix the tag types in the query without doing a UNION.

In SQL Server Management Studio, the extension tables are listed under the INSQL or INSQLD linked servers under the **Server objects** tree item.

History Table Format

The History table presents acquired plant data in a historical format, which is shown as follows:

DateTime	TagName	Value	vValue	Quality	QualityDetail	(continued...)
02:17:01:03	Temp1	78	78	0	192	(continued...)
02:17:01:03	Temp2	79	79	0	192	(continued...)
02:17:01:03	Temp3	77	77	0	192	(continued...)
02:17:01:03	Temp4	80	80	0	192	(continued...)
02:17:01:04	Temp1	77	77	0	192	(continued...)
02:17:01:04	Temp2	78	78	0	192	(continued...)
02:17:01:04	Temp3	76	76	0	192	(continued...)
02:17:01:04	Temp4	79	79	0	192	(continued...)
02:17:01:05	Temp1	76	76	0	192	(continued...)
02:17:01:05	Temp2	77	77	0	192	(continued...)
02:17:01:05	Temp3	78	78	0	192	(continued...)
02:17:01:05	Temp4	80	80	0	192	(continued...)

There is one row for a single tag's value for a particular timestamp.

Note The AnalogHistory, DiscreteHistory, and StringHistory tables are provided for backward compatibility and can only accept tagnames in the SELECT statement that are of the same type; that is, you cannot mix the tag types in the query without doing a UNION. The History table, however, can accommodate a mixture of tag types and should be used instead of the AnalogHistory, DiscreteHistory, or StringHistory tables. The Value column returns a float value for analog and discrete tags, a NULL for string tags. The vValue column returns a sql_variant for all tag types.

"Wide" History Table Format

The WideHistory table contains the same data as the History table, but in a different format. The WideHistory table presents data for one or more tag values for a single timestamp, thus providing a "wide" view of the data. To query for values in the WideHistory table, you must specify the timestamp and one or more tagnames as the column names in the query syntax. The results will contain a column for the timestamp and columns for the value of each specified tag at that timestamp. In the following example, **Temp1**, **Temp2**, **Temp3**, and **Temp4** are tagnames:

DateTime	Temp1	Temp2	Temp3	Temp4
02:17:01:03	78	79	77	80
02:17:01:04	77	78	76	79
02:17:01:05	77	78	76	79

Using the History table to perform the same task is much more difficult.

You can also specify search criteria for the values you want to return (for example, where Temp1 > 75). The WideHistory table can only be related to other tables based on the timestamp.

Note The AnalogWideHistory, DiscreteWideHistory, and StringWideHistory tables are provided for backward compatibility and can only accept tagnames in the SELECT statement that are of the same type; that is, you can't mix the tag types in the query. The WideHistory table, however, can accommodate a mixture of tag types and should be used instead of the AnalogWideHistory, DiscreteWideHistory, or StringWideHistory tables.

The WideHistory table column type returns a SQL Server type float for analog, a SQL Server type int for discrete tags, and an nvarchar(512) for string tags. The schema of the definition table, WideHistory_OLEDB, indicates a sql_variant type. This is simply a shorthand notation; it does not represent the type actually returned.

There is no Quality column for the WideHistory table because there is more than one tag value for each row returned. However, a value returned for a specified tag will be set to NULL if the quality of the value is invalid, inhibited, or unavailable.

The following restrictions apply when performing a query against the WideHistory table:

- Column names must be specified.
- The table is only accessible using the OPENQUERY statement.

Because tagnames are used for column names, the tagname can include any characters as defined by the rules for Microsoft SQL Server identifiers. An identifier that does not comply with the rules for the format of regular identifiers must always be delimited using brackets ([ ]). For more information on identifiers and delimiters, see your Microsoft SQL Server documentation.

If you include an illegal column name in your query and do not use delimiters, no data will be returned.

The StateWideHistory table is similar to the WideHistory table, except that it allows for retrieval of calculated "time in state" values for multiple tags, instead of actual history values. This table includes a vValue column, and the tag columns contain the time in state for the corresponding value. For more information on this table, see "StateWideHistory (INSQL.Runtime.dbo.StateWideHistory)" on page 95. For information on how to query this table, see "ValueState Retrieval" in Chapter 7, "Data Retrieval Options," in your *Wonderware Historian Concepts Guide*.

"Live" Table Format

The Live table presents the current value of the specified tag(s). The format of the Live table is as follows. The DateTime column will indicate the time the value was received.

DateTime	TagName	Value	vValue	Quality	QualityDetail	(continued...)
02:17:01:05	Temp1	77	77	0	192	(continued...)
02:17:01:05	Temp2	78	78	0	192	(continued...)
02:17:01:05	Temp3	76	76	0	192	(continued...)
02:17:01:05	Temp4	79	79	0	192	(continued...)

Note The AnalogLive, DiscreteLive, and StringLive tables are provided for backward compatibility and can only accept tagnames in the SELECT statement that are of the same type; that is, you can't mix the tag types in the query. The Live table, however, can accommodate a mixture of tag types and should be used instead of the AnalogLive, DiscreteLive, or StringLive tables.

Event Tables

Event tables contain definitions for events, including tags associated with events, detectors for events, and actions for events. The event system tables can also store "snapshots" of tag values at the time of an event, as well as details about the event itself.

A special type of event action is a summarization of tag values. A subset of the event tables provide the supporting framework for fully automated summary generation for analog, discrete and string tags. The event system tables are:

ActionType
AnalogSnapshot
CalcType
DetectorType
DiscreteSnapshot
EventHistory
EventTag
EventTagPendingDelete*
Frequency
SnapshotTag
SQLTemplate
StringSnapshot
SummaryData
SummaryOperation
SummaryHistory
SummaryTagList
Tag
TimeDetectorDetail
TimeDetectorDetailPendingDelete*

* System-level table. Do not edit.

InTouch Integration Tables

InTouch® integration tables contain information about tag definitions that are imported into the Runtime database from an InTouch application. The InTouch integration tables are:

InTouchNode	InTouchSpecific
TopicImportInfo	

Modification Tracking Tables

The modification tracking tables contain information about changes that are made to columns in the database. The modification tracking tables are:

History (INSQL.Runtime.dbo.History)	ModLogColumn
ModLogTable	UserDetail

Namespaces and Grouping Tables

The namespaces and grouping tables contain information that defines how sets of tags can be grouped together for alarming, displays, event management, and batch management. These tables also define hierarchies for items in the system, public, or private namespaces. The namespaces and grouping tables are:

PrivateGroupTag	PrivateNameSpace
PublicGroupTag	PublicNameSpace
ServerList	Tag
TagRef	UserDetail

Tag Definition Tables

Types of tags that can be defined in the Wonderware Historian are analog, discrete, event, and string. All tag types have some attributes that are the same, such as a tag name, a description, or a storage rate. These attributes are stored in the Tag table. Information specific to a particular tag type, such as maximum and minimum values, limits, and engineering units, is stored in related, type-specific tables. For example, the total definition for an analog tag would be partially stored in the Tag table and partially stored in the AnalogTag table.

The tag definition tables are:

AnalogSummaryTag	AnalogTag
Annotation	Context
Deviation	DiscreteTag
EngineeringUnit	EventTag
Limit	LimitName
Message	RateOfChange
StringTag	Tag
TagRef	Topic

Replication Tables

Tag information can be replicated from source, or tier 1, servers to replication, or tier 2, servers. Replication lets you consolidate and summarize information from separate servers to a single replication server so you can then perform analyses and run reports from the replication server on the consolidated data. You can summarize tags to capture analog or state values. You can also do a simple replication, which copies tag information directly without summarizing it. For more information, see Chapter 9, "Replication Subsystem," in your *Wonderware Historian Concepts Guide*.

The replication tables are:

AnalogSummaryTag	CustomReplicationSchedule
IntervalReplicationSchedule	ReplicationGroup
ReplicationSchedule	ReplicationScheduleType
ReplicationServer	ReplicationSyncRequest
ReplicationTag	ReplicationTagEntity
ReplicationType	

System Configuration Tables

All Wonderware Historian parameters are stored in system configuration tables. Parameters include information regarding the historian's physical nodes, site-specific configuration parameters, and parameters pertaining to the physical I/O equipment to which the system is connected. The system configuration tables are:

ConfigStatusPending *	ConfigStatusSnapshot*
ErrorLog	IODriver
IOServer	IOServerType
LocalizedText*	OPCQualityMap
QualityMap	SnapshotDetail
StorageLocation	StorageNode
SystemParameter	Tag
TimeZone*	Topic
UserDetail	

* System-level table. Do not edit.

ArchestrA Browsing Tables

The ArchestrA browsing tables store information required to support the browsing of the ArchestrA model view hierarchy by Wonderware Historian clients. The ArchestrA browsing tables are:

aaAreaData	aaAreaXML
aaAttributeData	aaAttributeDataPending
aaObjectData	aaObjectDataPending

These tables are for internal use only.

Chapter 2

Tables

All information regarding how the system is configured is stored in tables in the Runtime database. Event history, summary history, and summary data are also stored in SQL Server tables. You can view the details of all tables by using the Microsoft SQL Server Management Studio.

Note Tables that have been retained for backward compatibility are listed in Chapter 6, "Backward Compatibility Entities."

Accessing Information About a Database Table

To find out basic information about a database table, such as the columns, primary and foreign keys, and indexes, use the **sp_help** stored procedure from the Runtime database. For example:

```
sp_help AnalogTag
```

For information about referential integrity for a table, see the HistorianDatabaseSchema.PDF.

aaAreaData

Contains one row for each item in the latest ArchestrA Area data package.

The Area data hierarchy is sent from ArchestrA to the historian in the form of an XML data package. In addition to data about the Areas, this package also contains data about the Galaxy, WinPlatforms, AppEngines, and DeviceIntegration Objects.

Column	Data Type	Description
AreaKey	int NOT NULL	The unique identifier for the item in the Area data hierarchy.
Category	int NOT NULL	The type of the item in the Area data hierarchy. 0 = Galaxy; 1 = WinPlatform; 3 = AppEngine; 13 = Area; 11 = DDESuiteLinkClient, OPCClient or InTouchProxy; 24 = RedundantDIObject. All other values are reserved for future use.
AreaName	nvarchar(255) NOT NULL	The name of the item in the Area data hierarchy.
ContainedName	nvarchar(255) NULL	The contained name (if relevant) of the item in the Area data hierarchy.
(FK) ParentKey	int NOT NULL	The unique identifier for the parent item of this item. For the Galaxy item, this value is 0.

aaAreaXML

Contains a single row describing the latest Area data sent from ArchestrA.

Column	Data Type	Description
Version	bigint NULL	The version number of the latest ArchestrA Area data package.
AreaXML	ntext NULL	Reserved for future use.

aaAttributeData

Contains one row for each attribute referenced by an object in the ArchestrA namespace.

Column	Data Type	Description
AttributeName	nvarchar(256) NOT NULL	The ArchestrA attribute name. This name corresponds to a Wonderware Historian tagname.
(FK) ObjectKey	int NOT NULL	ObjectKey is a foreign key from the aaObjectData table.
wwDomainTagKey	int NOT NULL	The unique numerical identifier for the ArchestrA attribute (historian tag) in a specific domain.

aaAttributeDataPending

Contains one row for each attribute in the latest ArchestrA attribute data package.

Column	Data Type	Description
AttributeName	nvarchar(256) NOT NULL	The ArchestrA attribute name. This name corresponds to a Wonderware Historian tagname.
(FK) ObjectKey	int NOT NULL	ObjectKey is a foreign key from the aaObjectDataPending table.

aaObjectData

Contains one row for each object in the ArchestrA namespace.

Column	Data Type	Description
ObjectKey	int NOT NULL	The unique identifier for the object. This column does not have the same numeric value as ObjectKey column of the aaObjectDataPending table.
Type	int NOT NULL	The type of the object. 0 = Area; 1 = ApplicationObject (regular); 2 = Traceability object. All other values are reserved for future use.
aaTagName	nvarchar(256) NULL	The ArchestrA tag name for the object.

Column	Data Type	Description
ContainedName	nvarchar(256) NULL	The ArchestrA contained name for the object.
(FK) ParentKey	int NOT NULL	The unique identifier for the parent of this object.
Status	tinyint NOT NULL	Used to indicate whether a name change has occurred. 0 = No change; 1 = The tag name has changed; 2 = The contained name has changed. The default is 0.

aaObjectDataPending

Contains one row for each object in the latest ArchestrA object data package.

Column	Data Type	Description
ObjectKey	int NOT NULL	The unique identifier for the object. This identifier is unique only within an object data package and may be repeated in subsequent data packages.
Type	int NOT NULL	The type of the object. 0 = Area; 1 = ApplicationObject (regular); 2 = Traceability object. All other values are reserved for future use.
aaTagName	nvarchar(256) NULL	The ArchestrA tag name for the object.
ContainedName	nvarchar(256) NULL	The ArchestrA contained name for the object.
(FK) ParentKey	int NOT NULL	The unique identifier for the parent of this object.

ActionType

Contains one row for each type of event action.

Column	Data Type	Description
ActionTypeKey	int IDENTITY	The unique identifier for a particular type of action. Event tags and actions are linked by this key. The event subsystem relies on the following values, which are added during installation: 1 = No action; 2 = Generic SQL; 3 = Snapshot; 4 = E-mail; 5 = Deadband; 6 = Summary. This value is automatically generated when a new action is created.
Name	nvarchar(33) NOT NULL	The name given to the type of action.
Description	nvarchar(50) NULL	The description of the action.
EditorClassName	nvarchar(80) NULL	The name by which the component is referenced by a client application, such as the InSQL Console, in order to provide a visual representation.
ActionClassName	nvarchar(80) NULL	The name by which the action component (COM object) is referenced in the system in order to perform the action.

AnalogSnapshot

Contains one row for each analog tag value that was configured to be stored when a defined event occurred. To view analog, discrete, and string snapshot values at the same time, use the v_SnapshotData view instead. For more information, see "v_SnapshotData" on page 129.

Column	Data Type	Description
(FK) SnapshotTagKey	int NOT NULL	The unique numerical identifier of the tag included in the snapshot. SnapshotTagKey is a foreign key from the SnapshotTag table.
(FK) EventLogKey	int NOT NULL	The unique numerical identifier of an event occurrence. EventLogKey is a foreign key from the EventHistory table.
Value	float(8) NULL	The value of the tag at the time of the event occurrence. Measured in engineering units.

Column	Data Type	Description
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.
QualityDetail	int NULL	An internal representation of data quality.

AnalogSummaryHistory (INSQL.Runtime.dbo.AnalogSummaryHistory)

The AnalogSummaryHistory view returns results for analog summary points.

Column name	Type and nullability	Description
TagName	nvarchar(256) NOT NULL	The name of the summary tag.
StartDateTime	datetime/datetime2 (7) NOT NULL	Start time of the retrieval cycle for which this row is returned. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
EndDateTime	datetime/datetime2 (7) NOT NULL	End time of the retrieval cycle for which this row is returned. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)

Column name	Type and nullability	Description
OPCQuality	int NULL	OPC quality. Normal OPC quality retrieval logic is applied if: • All the point found and processed for this row have GOOD quality. If they all have the same GOOD quality, then that quality is returned. • If there is a gap in the entire calculation cycle, then BAD quality is returned for the tag. • For any other scenario with any mixture of GOOD and BAD points, a DOUBTFUL OPC quality (64) is returned. For more information, see "Quality Rule (wwQualityRule)" in Chapter 7, "Data Retrieval Options," in your *Wonderware Historian Concepts Guide*.
PercentGood	float(53) NULL	Time in seconds that the value was good for the retrieval cycle (pro-rated for partial cycles).
First	float(53) NULL	If at least one non-NULL point exists for the tag in question within the retrieval cycle, then the value returned is the first point stored with a time stamp within the retrieval cycle. If no points exist within the retrieval cycle, then the value returned is the current value at the cycle start time. If no non-NULL points can be found, then NULL is returned.
FirstDateTime	datetime/datetime2 (7) NULL	Timestamp associated with first value. This might be earlier than StartDateTime if this is the initial value for the retrieval cycle. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)

Column name	Type and nullability	Description
Last	float(53) NULL	If at least one non-NULL point exists for the tag in question within the retrieval cycle, then the value returned is the last point stored with a time stamp within the retrieval cycle. If no points exist within the retrieval cycle, then the value returned is the current value at the cycle start time. If no non-NULL points can be found, then NULL is returned.
LastDateTime	datetime/datetime2 (7) NULL	Timestamp associated with last value. This might be earlier than StartDateTime if this is the initial value for the retrieval cycle. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
Minimum	float(53) NULL	If at least one non-NULL point exists for the tag in question within the retrieval cycle, then the value returned is the minimum point stored with a time stamp within the retrieval cycle. If no points exist within the retrieval cycle, then the value returned is the current value at the cycle start time. If no non-NULL points can be found, then NULL is returned.
MinDateTime	datetime/datetime2 (7) NULL	Timestamp associated with Min value. NULL if Min is NULL. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
Maximum	float(53) NULL	If at least one non-NULL point exists for the tag in question within the retrieval cycle, then the value returned is the maximum point stored with a time stamp within the retrieval cycle. If no points exist within the retrieval cycle, then the value returned is the current value at the cycle start time. If no non-NULL points can be found, then NULL is returned.

Column name	Type and nullability	Description
MaxDateTime	datetime/datetime2 (7) NULL	Timestamp associated with Max value. NULL if Max is NULL. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
Average	float(53) NULL	Time weighted average value of retrieval cycle. This is calculated by using the individual summary averages. The calculation is "Sum(average * delta t) / Total time of average in all cycles" - delta t is prorated for any partially contained storage cycles For analog tags, the calculation is "Sum(value * delta t) / Total time. (This is like the values returned by an Average query against the History table for a cycle of the same length, where the History row DateTime is the same as the EndDateTime here.)
StdDev	float(53) NULL	Time weighted standard deviation value of the retrieval cycle. The value is calculated using time weighted sums (Integrals) and time weighted sums of squares (IntegralOfSquares) values, prorated for any partially contained storage cycles. For analog tags, similar StdDev values are produced for each cycle.
Integral	float(53) NULL	Area under value curve of retrieval cycle. The calculation is "Sum(value * delta t) / Total time of integral in all cycles" - delta t is prorated for any partially contained storage cycles For analog tags, the calculation is "Sum(value * delta t) / Total time. (This is like the values returned by an Integral query against the History table for a cycle of the same length, where the History row DateTime is the same as the EndDateTime here.) For analog tags, similar Integral values are produced for each cycle.
ValueCount	int NULL	Number of values in a particular cycle.
SourceTag	nvarchar(256) null	The source (tier 1) tag for the summary tag.

Column name	Type and nullability	Description
SourceServer	nvarchar(256) null	The source (tier 1) server for the summary tag.
wwCycleCount	int NULL	The number of cycles into which the entire query time range has been divided.
wwResolution	int NULL	Length of cycles in milliseconds. The default is 3600000 (equal to 1 hour).
wwTimeZone	nvarchar(50) NULL	Time zone to use for interpreting both input and output timestamp parameters. If none is specified, then the default is set to LOCAL.
wwVersion	nvarchar(30) NULL	Data version, ORIGINAL or LATEST. If none is specified, the default is LATEST.
wwTagKey	int NOT NULL	Tag key.
wwRetrievalMode	nvarchar(16) NOT NULL	Determines whether to use CYCLIC or DELTA retrieval. The default is DELTA.

AnalogSummaryTag

Contains one row for each defined analog summary tag. (This is used exclusively for tiered historian installations.) Configuration information specific to analog summary tags is stored in this table, while general information for all tag types is stored in the Tag table.

Column	Data Type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
(FK) EUKey	int NOT NULL	The unique numerical identifier of an engineering unit. EUKey is a foreign key from the EngineeringUnit table.
MinEU	float(25) NOT NULL	The minimum value of the tag, measured in engineering units.
MaxEU	float(25) NOT NULL	The maximum value of the tag, measured in engineering units.
MinRaw	float(25) NULL	The minimum value of the raw acquired value.

Column	Data Type	Description
MaxRaw	float(25) NULL	The maximum value of the raw acquired value.

AnalogTag

Contains one row for each defined analog tag. Configuration information specific to analog tags is stored in this table, while general information for all tag types is stored in the Tag table.

Column	Data Type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
(FK) EUKey	int NOT NULL	The unique numerical identifier of an engineering unit. EUKey is a foreign key from the EngineeringUnit table.
MinEU	float(25) NOT NULL	The minimum value of the tag, measured in engineering units.
MaxEU	float(25) NOT NULL	The maximum value of the tag, measured in engineering units.
MinRaw	float(25) NULL	The minimum value of the raw acquired value.
MaxRaw	float(25) NULL	The maximum value of the raw acquired value.
Scaling	int NOT NULL	The type of algorithm used to scale raw values to engineering units. For linear scaling, the result is calculated using linear interpolation between the end points. 0 = None; 1 = Linear; 2 = Square Root. (Square root is reserved for future use).
RawType	int NOT NULL	The numeric type for the raw value. 1 = Euro Float, an outdated data type (4 bytes); 2 = MS Float (4 bytes); 3 = Integer (2 or 4 bytes); 4 = MS Double (reserved for future use) (8 bytes).

Column	Data Type	Description
ValueDeadband	float(25) NOT NULL	The percentage of the difference between the minimum and maximum engineering units for the tag. Any data values that change less than the specified deadband are not stored. The value deadband applies to delta storage only. A value of 0 indicates that a value deadband will not be applied.
IntegerSize	tinyint NOT NULL	The bit size of the analog tag. 12 = 12-bit; 15 = 15-bit; 16 = 16-bit; 32 = 32-bit; 64 = 64-bit (reserved for future use).
SignedInteger	bit NOT NULL	Used to specify whether an integer is a signed number (positive or negative) or an unsigned number (positive only). 0 = Unsigned; 1 = Signed.
RateDeadband	float(25) NOT NULL	The percentage of deviation in the full-scale value range for an analog tag. The swinging door (rate) deadband applies to delta storage only. Time and/or value deadbands can be used in addition to the swinging door deadband. Any value greater than 0 can be used for the deadband. A value of 0 indicates that a swinging door deadband will not be applied. For more information on swinging door storage, see "Delta Storage" in Chapter 5, "Data Storage Subsystem," in your *Wonderware Historian Concepts Guide*.
InterpolationType	tinyint NOT NULL	The interpolation type for retrieval. 0 = Stair-stepped interpolation; 1 = Linear interpolation (if applicable, based on the tag type); 254 = System default interpolation mode. The system default interpolation type is to use the system default for the analog type, either integer or real. The system default interpolation type for an analog type is determined by the setting of the InterpolationTypeInteger and InterpolationTypeReal system parameters. This setting impacts Interpolated, Average, and Integral retrieval modes.

Column	Data Type	Description
RolloverValue	float NOT NULL	The first value that causes the counter to "roll over." This rollover value is used by the "counter" retrieval mode. For example, a counter that counts from 0 to 9999, the counter rolls over back to 0 for the 10,000th value it receives. Therefore, set the rollover value to 10,000.

Annotation

Contains one row for each user annotation about a tag value. Users can make personal (or public) notes about a tag value. This information is stored with the tag value and timestamp to which the annotation applies. Each annotation in this table is linked to a database user.

Column	Data Type	Description
AnnotationKey	int IDENTITY	The unique numerical identifier for the annotation. This value is automatically generated by the system when the annotation is added.
(FK) UserKey	int NOT NULL	The unique numerical identifier for a database user as defined in the UserDetail table. UserKey is a foreign key from the UserDetail table.
(FK) TagName	nvarchar(256) NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
DateCreated	datetime NULL	The date that the annotation was created.
Content	nvarchar(1000) NOT NULL	The annotation text.
DateTime	datetime NOT NULL	The timestamp of the tag value for which the user has made an annotation.
Value	float(25) NULL	The value of the tag at the time of the annotation.

AttributeType

Contains one row for each attribute type.

Column	Data Type	Description
AttributeTypeKey	int NOT NULL	The unique numerical identifier for the attribute. This value is automatically generated by the system when the attribute is added.
AttributeTypeName	nvarchar(255) NOT NULL	The name of the attribute type.
AttributeTypeValue	tinyint NOT NULL	The bit mask for the attribute type.

CalcType

Contains one row for each type of summary calculation that can be performed by the event subsystem.

Column	Data Type	Description
CalcType	CalcTypes/char(3) NOT NULL	The type of calculation to be performed: SUM, MAX, MIN, or AVG.
Description	nvarchar(50) NULL	The description of the calculation.

ConfigStatusPending

Contains one row for each database modification that requires a reinitialization of the system.

Important Do not edit this table.

Column	Data Type	Description
ID	int IDENTITY	The unique identifier for the database modification.
Type	tinyint NOT NULL	Used to indicate the type of object to which the modifications apply. 0 = IDAS; 1 = IOServer; 2 = Topic; 3 = Tag; 4 = StorageLocation; 5 = SnapshotDetail; 6 = NamedSystemParameter; 7 = EngineeringUnit.

Column	Data Type	Description
ObjectKey	int NOT NULL	The unique identifier of the modified object. If the modified object is a system parameter, the value will be 0. For all other object types, the value is from one of the following tables and columns: IODriver.IODriverKey; IOServer.IOServerKey; Topic.TopicKey; Tag.wwTagKey; StorageLocation.StorageType; SnapshotDetail.StorageSize.
Status	tinyint NULL	Used to indicate the type of modification. 1 = Insert; 2 = Update; 3 = Delete; 6 = The tag's source has changed (that is, if the value of the IOServerKey or TopicKey column in the Tag table has changed).

ConfigStatusSnapshot

When changes to the historian system are committed, a snapshot of the contents of the ConfigStatusPending table are stored to this table. The internal configuration object then finishes processing the reinitialization based on the data in this table, while any new changes are being stored in the ConfigStatusPending table.

The columns in this table are identical to the columns in the ConfigStatusPending table.

Important Do not edit this table.

Context

Contains one row for each context to which a group of limits, rates of change, or deviations can belong. Example contexts are "Normal Operation" and "Cold Shutdown."

Column	Data Type	Description
ContextKey	int IDENTITY	The unique numerical identifier for the context. This value is automatically generated when a new context is added.
Description	nvarchar(50) NOT NULL	The description of the context.

CurrentEditor

Contains one row for each editor.

Column	Data Type	Description
CurrentEditor	tinyint NOT NULL	Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server.
EditorName	nvarchar(max) NOT NULL	The name of the editor.

CustomReplicationSchedule

Contains one row for each trigger time for a custom replication schedule of ScheduleType CUSTOM. (This is used exclusively for tiered historian installations.) Interval-based replication schedules are handled in the IntervalReplicationSchedule table.

Column	Data Type	Description
(FK) ReplicationScheduleKey	int NOT NULL	The unique identifier for the schedule. ReplicationScheduleKey is a foreign key from the ReplicationSchedule table.

Column	Data Type	Description
TimeOfDay	nvarchar(10)	The time of day (in the local time for the Wonderware Historian) for the trigger time in the custom replication schedule. This value is automatically populated based on the schedule. The format is <Hour:Minutes><AM/PM>. Time is displayed on a 12-hour clock.

DetectorType

Contains one row for each type of event detector.

Column	Data Type	Description
DetectorTypeKey	int IDENTITY	The unique identifier of a particular type of detector. Event tags and detectors are linked by means of this key. The event system relies on the following values, which are added during installation: 1 = System; 2 = External event; 3 = Generic SQL; 4 = Analog specific value; 5 = Discrete specific value; 6 = Time-based (schedule). This value is automatically generated when a new detector is created.
Name	nvarchar(33) NOT NULL	The name given to the type of detector.
Description	nvarchar(50) NULL	The description of the detector.
EditorClassName	nvarchar(80) NULL	The name by which the component is referenced by a client application, such as the InSQL Console, in order to provide a visual representation.
DetectorClassName	nvarchar(80) NULL	The name by which the detector component (COM object) is referenced in the system in order to perform the detection.
ExecutionMode	tinyint NOT NULL	Used to specify the manner in which the detector executes. 0 = Executed cyclically by the event subsystem according to the event tag scan rate; 1 = Asynchronous and triggered by an external mechanism. The default is 0.

Deviation

Contains one row for each defined deviation for an analog tag. The deviation is the percentage of change in a tag's value from a fixed value, called the target. Each analog tag can have two defined deviations: major and minor. This table is populated when an InTouch application is imported and is not used by the Wonderware Historian.

Column	Data Type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
(FK) ContextKey	int NOT NULL	The unique numerical identifier for the context. ContextKey is a foreign key from the Context table.
MinorDeviation	real NULL	The percentage that the tag can deviate from the target value before a minor deviation alarm condition is produced.
MinorChecked	bit NOT NULL	Used to determine the alarm state of the tag based on the minor deviation. 0 = Not in an alarm condition; 1 = In an alarm condition.
MinorPriority	int NULL	The priority level for the minor deviation. Valid values are numbers between 1 and 999, with 1 being the highest priority and 999 being the lowest priority.
MajorDeviation	real NULL	The percentage that the tag can deviate from the target value before a major deviation alarm condition is produced.
MajorChecked	bit NOT NULL	Used to determine the alarm state of the tag based on the major deviation. 0 = Not in an alarm condition; 1 = In an alarm condition.
MajorPriority	int NULL	The priority level for the major deviation. Valid values are numbers between 1 and 999, with 1 being the highest priority and 999 being the lowest priority.
Target	float(8) NULL	The reference value of the tag from which minor and/or major deviation percentages are based.

Column	Data Type	Description
Deadband	real NULL	The deviation percentage the tag value must drop below the target before the tag is taken out of alarm.

DiscreteSnapshot

Contains one row for each discrete tag value that was configured to be stored when a defined event occurred. To view analog, discrete, and string snapshot values at the same time, use the v_SnapshotData view instead. For more information, see "v_SnapshotData" on page 129.

Column	Data Type	Description
(FK) SnapshotTagKey	int NOT NULL	The unique numerical identifier of the tag included in the snapshot. SnapshotTagKey is a foreign key from the SnapshotTag table.
(FK) EventLogKey	int NOT NULL	The unique numerical identifier of an event occurrence. EventLogKey is a foreign key from the EventHistory table.
Value	tinyint NULL	The state of the discrete tag at the time that the event occurred. 0 = FALSE; 1 = TRUE.
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.
QualityDetail	int NULL	An internal representation of data quality.

DiscreteTag

Contains one row for each defined discrete tag. Configuration information specific to discrete tags is stored in this table, while general information for all tag types is stored in the Tag table.

Column	Data Type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
(FK) MessageKey	int NOT NULL	The unique numerical identifier of a TRUE/FALSE message pair that can be associated with a discrete tag. MessageKey is a foreign key from the Message table.

EngineeringUnit

Contains one row for each defined engineering unit (unit of measure).

Column	Data Type	Description
EUKey	int IDENTITY	The unique numerical identifier of an engineering unit. This value is automatically generated by the system when the engineering unit is added.
Unit	nvarchar(32) NULL	The unit of measure. Examples are mph, grams, and pounds.
DefaultTagRate	int NULL	The default rate, in milliseconds, at which tags are cyclically stored, based on engineering units. Although the system does not make use of this engineering unit based tag rate, you can reference this value in custom SQL scripts. The value you enter for this tag rate does not affect the default storage rate set for the tag.
IntegralDivisor	float(25) NOT NULL	The factor to be applied when integrating a rate with the units [EngUnits/TimeUnit] to a quantity with units [EngUnits]. This factor is called the integral divisor. The default value of 1 assumes a time unit of seconds and ensures that a rate of [Unit/second] is correctly integrated to [Unit]. For a time unit of minutes, set the integral divisor value to 60; for a unit of hours, set the integral divisor value to 3600. The integral divisor is applied similarly to rates or quantities that are not expressed in terms of a time unit. For example, to convert watts to watt-hours, the integral divisor is 1/3600. To convert watts to kilowatt-hours, the integral divisor is 1/3600000.
Status	tinyint NULL	Automatically updated by the system if a change is made to the engineering unit: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.

ErrorLog

Contains one row for each system message (or error message). Typically, this table is not used. The actual message text is stored in the LocalizedText table, and can be retrieved by specifying the error code in the SQL query. Or, you can use the v_ErrorLog view to retrieve the data included in this table, plus the actual text.

Column	Data type	Description
DateTime	datetime NOT NULL	The date that the message was written to the system log, in the local time of the Wonderware Historian.
Type	nvarchar(10) NULL	The type of system message.
ErrorCode	int NULL	The unique identifier for the message.
Parameter	nvarchar(256) NULL	Optional details pertaining to the message text. For example, for the message "Disk space remaining on circular path" the parameter would contain the number of MB.
TotalCount	int NULL	Used to prevent "flooding" conditions in the log file. If a particular message is generated numerous times during a relatively short period of time, the message is written to the log file only once, and the total number of times that it occurred appears in this column.
ModuleID	int NULL	A unique number assigned to the Wonderware Historian subsystem that generated the message.
Host	nvarchar(32) NULL	The computer on which the Wonderware Historian subsystem runs.
FileName	nvarchar(64) NULL	Used to indicate the program file that contains the line of code that an error message comes from. Used for debugging.
Line	int NULL	Used to indicate the line of code that an error message comes from. Used for debugging.

EventHistory

Contains one row for each stored event, as labeled by the tagname. Event data must be configured to be logged into this table.

Column	Data Type	Description
EventLogKey	int IDENTITY	The unique numerical identifier of an event occurrence. This value is automatically generated by the system when the event record is added.
(FK) TagName	nvarchar(256) NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the EventTag table.
DateTime	datetime/datetime2 NOT NULL	The timestamp reflecting when event history data was acquired. This is the time when an event actually occurred. This time reflects the time zone of the Wonderware Historian. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
DetectDateTime	datetime/datetime2 NOT NULL	The timestamp reflecting when the event was detected by the event system. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
Edge	tinyint NULL	The "edge" for the event detection. 0 = Trailing; 1 = Leading; 2 = Both; 3 = None; 4 = Time Detector; 5 = External Detector.

EventTag

Contains one row for each event definition. Configuration information specific to event tags is stored in this table, while general information for all tag types is stored in the Tag table.

Column	Data Type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
(FK) DetectorTypeKey	int NULL	The unique identifier of a particular type of detector. Event tags and detectors are linked by means of this key. The event system relies on the following values, which are added during installation: 1 = System; 2 = External event; 3 = Generic SQL; 4 = Analog specific value; 5 = Discrete specific value; 6 = Time-based (schedule). DetectorTypeKey is a foreign key from the DetectorType table.
(FK) ActionTypeKey	int NULL	The unique identifier for a particular type of action. Event tags and actions are linked by this key. The event subsystem relies on the following values, which are added during installation: 1 = No action; 2 = Generic SQL; 3 = Snapshot; 4 = E-mail; 5 = Deadband; 6 = Summary. ActionTypeKey is a foreign key from the ActionType table.
ScanRate	int NULL	The interval, in milliseconds, at which the system checks to see if the event conditions specified by the detector occurred. This value must be greater than or equal to 500 milliseconds, and less than or equal to 1 hour (3600000 ms).
TimeDeadband	int NOT NULL	The minimum time, in milliseconds, between stored events. If more than one event occurs during the deadband, only the most recent will be stored. The system will not store another event until the specified time has elapsed. A time deadband of 0 indicates that the system will store all events. Reserved for future use.

Column	Data Type	Description
Logged	bit NOT NULL	Used to specify whether or not to log events for this tag into the EventHistory table. Event logging can only be turned off if no associated actions are configured. 0 = Not logged; 1 = Logged. The default is 1.
Status	tinyint NOT NULL	The flag used by the event system at system startup and during runtime to determine if the event tag has been modified. 0 = Posted. Any changes have been detected and effected by the system. 1 = New. An event tag has been inserted, but is not yet executing. 2 = Modification. An event tag has been updated, but the older one is already executing. 98 = Disabled. 99 = Disabling requested. The event tag does not execute, even though the definition still exists in the schema. Note that there may be a delay of up to 30 seconds before a change in an event tag is seen by the running system.
PostDetectorDelay	int NOT NULL	The amount of time, in milliseconds, that must elapse after an event is detected before the event action can be executed.
UseThreadPool	bit NOT NULL	Used to specify how system threads are used to process events. 1 = All events are handled by a single thread and a single logon to the SQL Server; 0 = Each event uses a separate system thread and logon. This will allow the event subsystem to manage the scan rates of each detector component concurrently. (Reserved for future use.)
DetectorString	nvarchar(1500) NULL	The script that contains the criteria for event detection. Detector scripts are executed on the local Wonderware Historian.
ActionString	nvarchar(1500) NULL	The script that specifies the event action. Action scripts run on the local Wonderware Historian.

Column	Data Type	Description
Priority	tinyint NOT NULL	The priority level for the action, either critical or normal. The priority level determines the sorting queue to which the action will be sent. The critical queue is used for highly important events. If a system overload condition occurs, events that are given a critical priority will always be processed first. Events that are given a normal priority will be processed after any critical events and may possibly be dropped (that is, not performed) on an overloaded system. 0 = Normal; 1 = Critical. The default is 0.
Edge	tinyint NOT NULL	The "edge" for the event detection. 0 = Trailing; 1 = Leading; 2 = Both; 3 = None; 4 = Time Detector; 5 = External Detector.

EventTagPendingDelete

Contains one row for each event tag that is pending deletion. This table is used internally by the system during the deletion process. The columns in this table are the same as in the EventTag table.

Frequency

Contains one row for each available frequency for summary operations.

Column	Data Type	Description
FrequencyID	int IDENTITY	The unique numerical identifier for the frequency. Used to link a frequency with a time-based detector. 1= Hourly; 2 = Daily; 3 = Weekly; 4 = Monthly; 5 = Periodic; 6 = Other (Reserved for future use). This value is automatically generated by the system when the summarized tag is added.
Frequency	nvarchar(12) NOT NULL	The name for the frequency.

History (INSQL.Runtime.dbo.History)

Contains one row for each stored tag value.

Column	Data Type	Description
DateTime	datetime/datetime2 (7) NOT NULL	The timestamp of the returned value. For delta retrieval, this is typically the time at which the value was acquired by the Wonderware Historian. For cyclic retrieval, this is the specific time requested or calculated (using a SQL function). If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
Value	float(53) NULL	The value of the tag at the timestamp. The value is always NULL for string tags.
vValue	nvarchar(4000) NULL	The value of the analog, discrete, or string tag stored as a sql_variant. Using this column in a query allows you to have values with mixed datatypes as a result.
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.
QualityDetail	int NULL	An internal representation of data quality.
OPCQuality	int NULL	The quality value received from the data source.
wwTagKey	int NOT NULL	The unique numerical identifier of a tag within a single Wonderware Historian.

Column	Data Type	Description
wwRowCount	int NULL	The number of rows to be returned for a specified time period. For cyclic retrieval, the rows are spaced evenly across the time period, and the default row count is 100 rows. For cyclic retrieval, the row count is applied for each tag in a query. This parameter has been deprecated; do not use. Use the wwCycleCount parameter instead.
wwResolution	int NULL	The sampling rate, in milliseconds, for retrieving the data in cyclic mode. The system returns values stored over the requested time period at the interval specified by the resolution. For example, if you specify a 5000 ms resolution, the system queries for all data during the time period and then only returns those values that occur at each 5000 ms interval, starting with the start date and ending with the end date.
wwEdgeDetection	nvarchar(16) NULL	The type of edge detection result set that the query will return. Valid values are NONE, LEADING, TRAILING, and BOTH.

Column	Data Type	Description
wwRetrievalMode	nvarchar(16) NULL	Used to specify how retrieved data is processed before it is returned to the client. Valid values are: CYCLIC, DELTA, FULL, INTERPOLATED, BESTFIT, AVERAGE, MINIMUM, MAXIMUM, INTEGRAL, SLOPE, COUNTER, VALUESTATE, and ROUNDTRIP. FULL = All stored values are returned; CYCLIC = All stored data for tags during the specified time interval are returned for the number of retrieval cycles or resolution specified; DELTA = Only values that changed during the specified time interval are returned. For all other modes, a calculation is performed by the system on the data and the value(s) are returned. The default is CYCLIC for retrieval from analog tables, DELTA for retrieval from discrete and string tables, and default is DELTA for retrieval from the History table, unless the specific retrieval mode implies otherwise. For example, SLOPE always has DELTA characteristics.
wwTimeDeadband	int NULL	The minimum time, in milliseconds, between returned values for a single tag. Applies only to delta retrieval.
wwValueDeadband	float(53) NULL	The percentage of full scale (range), in engineering units. Any value changes that are less than this percentage are not returned. Applies only to delta retrieval. The default is 0.

Column	Data Type	Description
wwTimeZone	nvarchar(50) NULL	Used to specify the time zone for retrieval. By default, the retrieval subsystem converts the UTC (Universal Time Coordinated) timestamps for the stored data to the local time of the Wonderware Historian computer, including adjustments for daylight savings time. To keep the timestamps in UTC, set the value of wwTimeZone to UTC. To convert the timestamps to a client computer's time zone, set this parameter to the appropriate time zone text key from the TimeZone table.
wwVersion	nvarchar(30) NULL	If the original data values have been modified in the database, use this column to specify which version of the stored data is to be retrieved. Valid values are: ORIGINAL or LATEST. If no parameter is specified, the latest version of the data is retrieved by default. Modification is indicated by the QualityDetail.
wwCycleCount	int NULL	The number of retrieval cycles (sub-intervals) for the specified time period. The cycles will be spaced evenly across the time period. For example, if you specify a cycle count of four, the time period will be divided into four even cycles, and one or more values (depending on the retrieval mode) will be returned per cycle.
wwTimeStampRule	nvarchar(20) NULL	Used to specify whether cyclic results are timestamped at the beginning of the cycle or at the end of the cycle. Valid values are START and END. If no timestamp rule is specified in the query, then retrieval uses the setting of the TimeStampRule system parameter.

Column	Data Type	Description
wwInterpolationType	nvarchar(20) NULL	Used to determine which analog value to return at a given cycle boundary. Valid values are STAIRSTEP and LINEAR. If STAIRSTEP is specified, no interpolation occurs. The last known point is returned with the given cycle time. If no valid value can be found, a NULL is returned. If LINEAR is specified, the system calculates a new value at the given cycle time by interpolating between the last known value prior to the cycle time and the first value after the cycle time.
wwQualityRule	nvarchar(20) NULL	Used to specify whether values with certain characteristics are explicitly excluded from consideration by data retrieval. This parameter will override the setting of the QualityRule system parameter. Valid values are GOOD, EXTENDED, or OPTIMISTIC. A quality rule of GOOD means that data values with doubtful (64) OPC quality will not be used in the retrieval calculations and will be ignored. Values with bad QualityDetail indicate gaps in the data. A quality rule of EXTENDED means that data values with both good and doubtful OPC quality will be used in the retrieval calculations. Values with bad QualityDetail indicate gaps in the data. A quality rule of OPTIMISTIC means that calculations that include some good and some NULL values will not cause the overall calculations to return NULL. You can apply wwQualityRule to all retrieval modes.

Column	Data Type	Description
wwStateCalc	nvarchar(20) NULL	Used to indicate the type of calculation to return in the StateTime column for the "value state" retrieval mode. Valid values are: MINIMUM, MAXIMUM, AVERAGE, TOTAL, or PERCENT. You can also use the shortened versions: MIN, MAX, AVG, or SUM. The default for this column is TOTAL.
StateTime	float(53) NULL	The amount of time in the state, expressed as a float (64-bit) number of milliseconds, for all time-in-state modes except for "Percent." For a time-in-state percentage calculation, this value is the percentage of the total time interval, in the range 0.0 to 100.0, that the value was in the state.
PercentGood	float(53) NULL	The ratio of the number of rows that have "good" quality to the total number of rows in the retrieval cycle, expressed as a percentage in the range 0 to 100.
wwParameters	nvarchar(128) NULL	Reserved for future use.
StartDateTime	datetime/datetime2 NOT NULL	Start time of the retrieval cycle for which this row is returned. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
SourceTag	nvarchar(256) NULL	Returns the name of the source tag for a replicated tag at the time this point was stored. With the SourceServer, this column uniquely identifies the tag from which this replicated point is coming.
SourceServer	nvarchar(256) NULL	Returns the name of the server from which replication occurred for this replicated tag at the time this point was stored.

Column	Data Type	Description
wwFilter	nvarchar(512) NOT NULL	The name of the filter. Filters are specified as C-like functions and parentheses are always required, even when the filter does not override the default parameters (no parameters are passed). Filter values are NoFilter, ToDiscrete(), SigmaLimit(), and SnapTo(). The default value is NoFilter. If the query does not specify the wwFilter element at all, or if its default value is not overridden, then no filter is applied.
wwValueSelector	nvarchar(20) NULL	Used to specify which column to return for specified analog summary tags in the four basic retrieval modes: DELTA, FULL, CYCLIC, and INTERPOLATED. The defined set of selectors are AUTO (the default in all modes if not overridden), MINIMUM or MIN, MAXIMUM or MAX, FIRST, LAST, AVERAGE or AVG, INTEGRAL, and STDDEV or STANDDEVIATION. The default AUTO setting returns the Last attribute in the Value column (which makes it accessible in the WideHistory table). You can only override the selector for the basic retrieval modes. FIRST, LAST, MIN, and MAX each have their own timestamp that will be used for the time part of the VTQ. AVG, INTEGRAL and STDDEV represent values that hold for the entire cycle so the summary period start time will be used for the time part of a VTQ.
wwMaxStates	int NULL	For internal use only.

HistoryBlock (INSQL.Runtime.dbo.HistoryBlock)

Contains one row for each block of history data stored on a storage node.

Column	Data Type	Description
FromDate	datetime/datetime2 (7) NOT NULL	The starting timestamp for the history block. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
ToDate	datetime/datetime2 (7) NOT NULL	The ending timestamp for the history block. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
StorageNodeKey	int NOT NULL	The unique numerical identifier for the storage node.
Description	nvarchar(50) NULL	The description of the history block.
OnLine	tinyint NOT NULL	Used to indicate whether or not the tag information for the history block is loaded into memory. 0 = Not loaded; 1 = Loaded.
HistoryArchived	int NOT NULL	Used to indicate whether or not the history block has been archived (backed up). 1 = No status; 2 = Archived; 3 = Restored; 4 = Deleted. Reserved for future use.
SummaryArchived	int NOT NULL	Used to indicate whether or not the tag summary has been archived (backed up). 1 = No status; 2 = Archived; 3 = Restored; 4 = Deleted. Reserved for future use.
EventArchived	int NOT NULL	Used to indicate whether or not the event has been archived (backed up). 1 = No status; 2 = Archived; 3 = Restored; 4 = Deleted. Reserved for future use.

Column	Data Type	Description
StorageAreaType	int NOT NULL	The paradigm used for storage. 1 = Circular; 2 = Alternate; 3 = Buffer; 4 = Permanent. Reserved for future use.
ArchiveDate	datetime/datetime2 (7) NULL	The date at which the history block was archived. Reserved for future use.
ArchiveLocation	nvarchar(50) NULL	The location to which the history block was archived. Reserved for future use.
Version	int NULL	The version number for the history block. 1 = Block format used until release 3.0; 2 = Block format used for releases 3.0 and later. Reserved for future use.
Compression	int NULL	The version number for cyclic compression. 1 = No compression; 2 = Huffman encoding. Reserved for future use.
Sequence	int NOT NULL	The sequence number for the data stream. (1...n) Reserved for future use.
TimeZoneOffset	int NULL	The UTC offset, in minutes, from the local timestamp for when the history block was created. For example, a value of 480 would indicate an 8-hour offset from UTC, which would be Pacific Standard Time.
wwTimeZone	nvarchar(50) NULL	Used to specify the time zone for retrieval. By default, the retrieval subsystem converts the UTC (Universal Time Coordinated) timestamps for the stored data to the local time of the Wonderware Historian computer, including adjustments for daylight savings time. To keep the timestamps in UTC, set the value of wwTimeZone to UTC. To convert the timestamps to a client computer's time zone, set this parameter to the appropriate time zone text key from the TimeZone table.

HistorianSysObjects

Contains one row for each object in the database for which changes can be tracked.

Column	Data Type	Description
id	int NOT NULL	The unique identifier for the object.
Type	char(2) NULL	The type of object. C = CHECK constraint; D = Default or DEFAULT constraint; F = FOREIGN KEY constraint; K = PRIMARY KEY or UNIQUE constraint; L = Log; P = Stored procedure; R = Rule; RF = Stored procedure for replication; S = System table; TR = Trigger; U = User table; V = View; X = Extended stored procedure. Currently, only changes for the user tables (object type U) are tracked.
Name	varchar(50) NULL	The name of the modified object.

IntervalReplicationSchedule

Contains one row for each replication schedule of ScheduleType INTERVAL. (This is used exclusively for tiered historian installations.) Custom replication schedules are handled in the CustomReplicationSchedule table.

Column	Data Type	Description
(FK) ReplicationScheduleKey	int NOT NULL	The unique identifier for the schedule. ReplicationScheduleKey is a foreign key from the ReplicationSchedule table.
Period	smallint NOT NULL	The period value.
Unit	nvarchar(32) NOT NULL	The name of the unit.

InTouchNode

Contains one row for each InTouch node from which a tagname data dictionary (Tagname.x) is imported into the Wonderware Historian.

Column	Data Type	Description
NodeKey	int NOT NULL	The unique numerical identifier of the named InTouch node. A node key is automatically generated by the system when a node is added.
MachineName	nvarchar(255) NOT NULL	The name of the computer on which the InTouch application resides.
ApplicationName	nvarchar(32) NULL	The name of the InTouch application (VIEW).
Path	nvarchar(250) NULL	The UNC path to the InTouch Tagname.X file.
Description	nvarchar(50) NULL	The description of the InTouch node.
DuplicateChar	nvarchar(12) NOT NULL	The string that was added to a tag name as a prefix or suffix to make it unique.
PrefixOrSuffix	bit NOT NULL	Used to indicate whether unique tags were created by prefixing or suffixing the unique string for the node. 0 = Suffix; 1 = Prefix. Internal use only.
AlwaysModifyName	bit NOT NULL	Used to indicate whether a uniqueness string was added to every tag for the node. Internal use only.
ImportPlantTags	tinyint NOT NULL	Used to indicate whether plant tags were imported. (In InTouch, plant tags are called I/O tags.) Internal use only.
ImportSystemTags	tinyint NOT NULL	Used to indicate whether system tags were imported. Internal use only.
ImportMemoryTags	tinyint NOT NULL	Used to indicate whether memory tags were imported. Internal use only.
ImportAllTags	int NOT NULL	Used to indicate whether all tags were imported. Internal use only.
FixedStorageRate	tinyint NOT NULL	The cyclic storage rate, in seconds, for imported tags. Internal use only.

Column	Data Type	Description
ImportRoute	tinyint NOT NULL	Used to indicate the type of import that was last performed for the node. Internal use only.

InTouchSpecific

Contains one row of import-related information for each data dictionary (Tagname.x) imported from InTouch HMI software.

Column	Data Type	Description
(FK) NodeKey	int NOT NULL	The unique numerical identifier of the named InTouch node. NodeKey is a foreign key from the InTouchNode table.
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
OriginalName	nvarchar(32) NOT NULL	The original tag name in an InTouch application. The tag name may be different than the Wonderware Historian tag name if a new name was generated to ensure uniqueness.
TypeInfo	int NOT NULL	The type of tag in an InTouch application. For more information about InTouch tag types, see your InTouch documentation. Internal use only.
InInSQL	bit NOT NULL	Used to specify whether or not the tag information has been imported from InTouch into the Wonderware Historian database. Internal use only.
Comment	nvarchar(50) NULL	The original description for the tag that was imported from InTouch.

IODriver

Contains one row for each IDAS providing data to the Wonderware Historian.

Column	Data Type	Description
IODriverKey	int IDENTITY	The unique identifier for an IDAS. This value is automatically generated by the system when the IDAS is added.
(FK) StorageNodeKey	int NOT NULL	The unique numerical identifier for the storage node. StorageNodeKey is a foreign key from the StorageNode table.
ComputerName	nvarchar(255) NOT NULL	The name of the computer on which the IDAS runs.
AltComputerName	nvarchar(255) NULL	The name of the computer on which an optional, redundant IDAS runs. You must use the fully qualified name of the computer. You could also use the IP address. This should be set to an empty string if no redundant IDAS is specified. Make sure that the IDAS software is installed on the target failover computer. If the failure of the primary IDAS is detected by the system, the failover IDAS is automatically started. The failover IDAS is shut down after the primary IDAS is back online. By default, this column is an empty string.

Column	Data Type	Description
StoreForwardMode	tinyint NOT NULL	Used to specify whether or not store-and-forward capability is enabled. If enabled, and the network connection between the IDAS and the storage node fails, data will be "buffered" to the location specified by the store-and-forward path. Valid values are: 0 = Disabled; 1 = Enabled; 2 = Autonomous. The Autonomous mode (2) is an extension of the normal store-and-forward mode (1). It allows the IDAS to start up using an IDAS configuration file and collect data in store-and-forward mode if the network connection to the Wonderware Historian is not available.

Column	Data Type	Description
StoreForwardPath	nvarchar(255) NULL	Used to specify the path for the IDAS data buffer on the local hard drive of the IDAS computer. The path should be absolute (for example, c:\IDASBuffer). Data is written to this path until the minimum threshold for the buffer is reached. Remote buffer paths are not supported. When the store-and-forward path specified for the IDAS is invalid, the default path picked by the system is: <public folder>\ArchestrA\Historian\IDAS\SF where the <public folder> is dependent on the operating system. For example, for the Windows 2008 and Windows Vista operating systems, the path is C:\ProgramData\ArchestrA\Historian\IDAS\SF. On the Windows 2003 and Windows XP operating systems, the path is C:\Documents and Settings\All Users\Application Data\ArchestrA\Historian\IDAS\SF. When the store-and-forward path specified for the IDAS is just a folder name (without any path characters like \ and :), the default path picked by the system is: <Windows system path>\<folder name specified by the user>. For example, for the Windows Server 2003, Windows XP, Windows Vista 32-bit, and Windows Server 2008 32-bit operating systems, the path is C:\WINDOWS\system32\<folder name>.
MinMBThreshold	int NOT NULL	The minimum amount of free disk space, in megabytes, at which IDAS stops collecting data in the store-and-forward buffer.

Column	Data Type	Description
Status	tinyint NULL	Automatically updated by the system if a change is made to IDAS: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.
Enabled	bit NOT NULL	Used to indicate whether the IDAS is enabled or not. 0 = Not enabled; 1 = enabled. Disabling the IDAS allows for the configuration to be retained in the database, even though the IDAS is removed from the system.
StoreForwardDuration	int NOT NULL	The minimum duration, in seconds, for the IDAS to function in store-and-forward mode. The IDAS functions in store-and-forward mode for this length of time even if the condition that caused IDAS to function in store-and-forward mode no longer exists. The maximum duration is 3600 seconds, and the minimum is 0 seconds.
AutonomousStartupTimeout	int NOT NULL	The amount of time, in seconds, that the autonomous IDAS should wait for configuration commands when started by the Configuration service before going to the autonomous mode. This timeout may need to be increased only if you have a large number of IDASs configured as autonomous on a slow network.
BufferCount	int NOT NULL	The number of 64 KB buffers pre-allocated for buffering data. This number may need to be increased to accommodate high data rates.

Column	Data Type	Description
FileChunkSize	int NOT NULL	The size, in bytes, of the data "chunks" that are sent to the historian when store-and-forward data is forwarded. The size of the chunks can be decreased to accommodate slower networks. Decrease this number only if the forwarding delay is greater than zero.
ForwardingDelay	int NOT NULL	The time interval, in milliseconds, at which "chunks" of store-and-forward data are forwarded to the historian. The length of the interval may need to be increased to accommodate slower networks.
ConnectionTimeout	int NOT NULL	The amount of time, in seconds, that the Configuration service attempts to communicate with an IDAS for configuration/reconfiguration. If this timeout elapses, the Configuration service assumes that the IDAS connection has been dropped. This number may need to be increased to accommodate slower networks.

IOServer

Contains one row for each I/O Server providing data to the Wonderware Historian.

Column	Data Type	Description
IOServerKey	int IDENTITY	The unique numerical identifier for the I/O Server. This value is automatically generated by the system when the I/O Server is added.
(FK) StorageNodeKey	int NOT NULL	The unique numerical identifier for the storage node. StorageNodeKey is a foreign key from the StorageNode table.
(FK) IODriverKey	int NULL	The unique identifier for an IDAS. IODriverKey is a foreign key from the IODriver table.

Column	Data Type	Description
(FK) ApplicationName	nvarchar(32) NULL	The application name of the I/O Server. This name is usually the same as the executable file name. ApplicationName is a foreign key from the IOServerType table.
Path	nvarchar(255) NULL	The full UNC path (including the filename) to locate the executable file for the I/O Server. If the I/O Server type key is specified, the filename may be omitted.
ComputerName	nvarchar(255) NULL	The name of the computer on which the I/O Server runs.
AltComputerName	nvarchar(255) NULL	The name of the computer on which an optional, failover I/O Server runs. The failover I/O Server must be running in order for the switch to be made.
AutoStart	bit NOT NULL	Used to control how the I/O Server starts up. 0 = Automatic startup when the system starts. 1 = Manual startup required. Currently not used.
ExeType	int NOT NULL	The type of executable for the I/O Server. Used by the Historian System Management Console to determine how to start the I/O Server. 0 = Service; 1 = Console application; 2 = Windows application.
InitializationStatus	tinyint NOT NULL	A control flag used to ensure that each I/O Server has been asked for the data type (integer or real) of each tag that it will send. Only needed after a database modification.
ProtocolType	tinyint NOT NULL	The protocol used by the Wonderware Historian server to communicate with the I/O Server. 1 = DDE; 2 = SuiteLink™; 3 = Wonderware Historian named pipe driver (for compatibility with IndustrialSQL Server 3.0 and previous versions). Of the operating systems currently supported by the Wonderware Historian, DDE is only supported on the Windows XP operating system.
Description	nvarchar(50) NULL	The description of the I/O Server.

Column	Data Type	Description
Status	tinyint NULL	Automatically updated by the system if a change is made to the I/O Server: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.

IOServerType

Contains one row for every known I/O Server type. Information about a new I/O Server is added to this table when a server is installed. This table is populated with the latest information about Wonderware I/O Servers at the time of shipping.

Column	Data Type	Description
ApplicationName	nvarchar(32) NOT NULL	The application name of the I/O Server. This name is usually the same as the executable file name.
Description	nvarchar(50) NULL	The description of the I/O Server type.
ExeName	nvarchar(255) NULL	The name of the I/O Server's executable file.
Revision	nchar(20) NULL	The revision number for the I/O Server.

Limit

Contains one row for each monitored limit for a specified tag. A limit can be associated with one or more tags and/or contexts.

Column	Data Type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the AnalogTag table.
(FK) ContextKey	int NOT NULL	The unique numerical identifier for the context. ContextKey is a foreign key from the Context table.
LimitType	tinyint NOT NULL	The type of limit; that is, whether it is a rising (up) or falling (down) limit. 0 = Rising; 1 = Falling.

Column	Data Type	Description
Value	float(8) NOT NULL	The value that is used as a specific limit for a tag. In theory, a tag can have an infinite number of limits defined.
(FK) LimitNameKey	int NOT NULL	The unique numerical identifier associated with a limit name. LimitNameKey is a foreign key from the LimitName table.
Priority	int NOT NULL	The priority for the limit. Priorities can range from 1 to over 2 billion, with 1 being the highest priority.
Checked	bit NOT NULL	Used to specify whether a tag imported from InTouch is configured for automatic limit checking. Only checked limits are imported. 0 = Checking disabled; 1 = Checking enabled.
Description	nvarchar(50) NULL	The description of the limit.

LimitName

Contains one row for each name that is associated with a defined limit. Examples are "high," "low," and "maintenance."

Column	Data Type	Description
LimitNameKey	int IDENTITY	The unique numerical identifier associated with a limit name. This value is automatically generated by the system when a limit is added.
Name	nvarchar(20) NULL	The name for the limit.

Live (INSQL.Runtime.dbo.Live)

Contains one row for each analog, discrete, or string tag. The value of each tag in this table is updated every time a new value is received.

Column	Data Type	Description
DateTime	datetime/datetime2 (7) NOT NULL	The timestamp reflecting when the data last changed. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
Value	float(53) NULL	The value of the tag at date/time. This value is always NULL for string tags.
vValue	sql_variant NULL	The value of the analog, discrete, or string tag stored as a sql_variant. Using this column in a query allows you to have values with mixed datatypes as a result.
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.
QualityDetail	int NULL	An internal representation of data quality.
OPCQuality	int NULL	The quality value received from the data source.
wwTagKey	int NOT NULL	The unique numerical identifier of a tag within a single Wonderware Historian.
wwRetrievalMode	nvarchar(16) NULL	For queries against this table, the value of this column is ignored.
wwTimeDeadband	int NULL	For queries against this table, the value of this column is ignored.
wwValueDeadband	float(53) NULL	For queries against this table, the value of this column is ignored.

Column	Data Type	Description
wwTimeZone	nvarchar(50) NULL	Used to specify the time zone for retrieval. By default, the retrieval subsystem converts the UTC (Universal Time Coordinated) timestamps for the stored data to the local time of the Wonderware Historian computer, including adjustments for daylight savings time. To keep the timestamps in UTC, set the value of wwTimeZone to UTC. To convert the timestamps to a client computer's time zone, set this parameter to the appropriate time zone text key from the TimeZone table.
wwParameters	nvarchar(128) NULL	Used for additional parameters that can be specified. By default, the value of this parameter is an empty string.
SourceTag	nvarchar(256) NULL	Returns the name of the source tag for a replicated tag at the time this point was stored. With the SourceServer, this column uniquely identifies the tag from which this replicated point is coming.
SourceServer	nvarchar(256) NULL	Returns the name of the server from which replication occurred for this replicated tag at the time this point was stored.
wwValueSelector	nvarchar(20) NULL	Used to specify which column to return for specified analog summary tags in the four basic retrieval modes, DELTA, FULL, CYCLIC, and INTERPOLATED. The defined set of selectors are AUTO (the default in all modes if not overridden), MINIMUM or MIN, MAXIMUM or MAX, FIRST, LAST, AVERAGE or AVG, and INTEGRAL. The default AUTO setting returns the Last attribute in the Value column (which makes it accessible in the WideHistory table). You can only override the selector for the basic retrieval modes.

LocalizedText

Contains one row for each string of text that can be returned to a client from Wonderware Historian (for example, error messages and status messages).

If you add new text to the LocalizedText table, you must stop and restart Wonderware Historian in order for the changes to go into effect.

TextKey	LanguageID	LocalizedText
TextKey	int NOT NULL	The unique identifier for the message.
LanguageID	int NOT NULL	The locale ID for the language used. This ID is also used in the SQL Server syslanguages table.
LocalizedText	nvarchar(256) NULL	The content of the message.

Message

Contains one row for each on/off message pair that can be associated with a discrete tag. For example, a message pair may be "Open" and "Closed" and could be associated with valve and switch positions.

Column	Data Type	Description
MessageKey	int IDENTITY	The unique numerical identifier of a TRUE/FALSE message pair that can be associated with a discrete tag. This value is automatically generated by the system when the message pair is added.
Message0	nvarchar(64) NULL	The message associated with the FALSE state of the discrete tag. The maximum number of characters is 64. A discrete tag set to 0 is in the FALSE state.
Message1	nvarchar(64) NULL	The message associated with the TRUE state of the discrete tag. The maximum number of characters is 64. A discrete tag set to 1 is in the TRUE state.

ModLogColumn

Contains one row for each database column on which an INSERT, UPDATE, or DELETE has been performed.

Column	Data Type	Description
(FK) ModTableKey	int NOT NULL	The unique numerical identifier for the modification. ModTableKey is a foreign key from the ModLogTable table.
ColumnName	nvarchar(30) NOT NULL	The name of the modified column.
OldValue	sql_variant NULL	The value stored in the column before the modification was made, if the modification was to a configuration table. For modifications to history data using SQL INSERT and UPDATE statements, this column contains the timestamp of the earliest data affected by the INSERT or UPDATE operation. If multiple changes are made to the same data, then only the most recent change will be contained in this column. This column is not used for modifications made to history data using a CSV file.
NewValue	sql_variant NULL	The new value stored in the column, if the modification was to a configuration table. For modifications to history data, this column contains the total count of consecutive value updates attempted.

For more information on modification tracking, see "Modification Tracking" in Chapter 2, "System-Level Functionality," in your *Wonderware Historian Concepts Guide*.

ModLogTable

Contains one row for each database table on which an INSERT, UPDATE, or DELETE has been performed.

Column	Data Type	Description
ModTableKey	int IDENTITY	The unique numerical identifier for the modification. This value is automatically generated by the system when a new modification record is added.

Column	Data Type	Description
(FK) id	int NOT NULL	The unique identifier for the object that was modified. id is a foreign key from the HistorianSysObjects table.
ModType	char(1) NOT NULL	The type of modification. U = Update; I = Insert; D = Delete; 1 = SQL insert; 2 = SQL original insert; 3 = SQL update; 4 = CSV insert; 5 = CSV original insert; 6 = CSV update; 7 = CSV multi-point update; 8 = CSV "fast load" insert.
RowKey	sql_variant NOT NULL	The key identifier for the column modified in the table. For example, TagName for the Tag table, Name for the Topic table, and so on.
UserKey	int NOT NULL	The unique numerical identifier for a database user as defined in the UserDetail table. This value is from the UserDetail table. Currently not used.
DateTime	datetime NOT NULL	The timestamp of when the modification occurred.
UserName	nvarchar(256) NOT NULL	The name of the database user that made the modification. The value of this column reflects the Windows authentication user name (for example, DOMAIN\user_login_name) or the SQL Server authentication user name (for example, dbo), depending on how the user is logged into the SQL Server when the modification is made. In the case of a CSV file import, this column contains the user name as it appears in the CSV file.

For more information on modification tracking, see "Modification Tracking" in Chapter 2, "System-Level Functionality," in your *Wonderware Historian Concepts Guide*.

OPCQualityMap

Contains one row for each defined OPC quality.

Column	Data Type	Description
OPCQuality	tinyint NOT NULL	The quality value received from the data source.
Description	nvarchar(100) NULL	The text that describes what the OPC quality value means. Do not modify this description.

PrivateGroupTag

Contains one row for each instance of a tag in a user's private namespace.

Column	Data Type	Description
(FK) NameKey	int NOT NULL	The unique identifier for the object in the namespace. NameKey is a foreign key from the PrivateNameSpace table.
(FK) UserKey	int NOT NULL	The unique numerical identifier for a database user as defined in the UserDetail table. UserKey is a foreign key from the UserDetail table.
(FK) wwDomainTagKey	int NOT NULL	The unique numerical identifier for a tag in a specific domain. wwDomainTagKey is a foreign key from the TagRef table.

PrivateNameSpace

Contains one row for each object in the private namespace. Objects in the private namespace can include plant machines, areas, tags, and so on, and are organized in a hierarchy. Allows for more than one name to map to a single tag.

Column	Data Type	Description
(FK) UserKey	int NOT NULL	The unique numerical identifier for a database user as defined in the UserDetail table. UserKey is a foreign key from the UserDetail table.
NameKey	int IDENTITY	The unique identifier for the object in the namespace. This value is automatically generated by the system when the object is added.
Type	int NULL	The value that specifies the type of namespace. 1 to 6 = Tag; 1 to 2 million = System; 2+ million = Groups. Within the system range, the following values designate ArchestrA object types: 1999023 = Galaxy; 1999001 = WinPlatform object; 1999003 = AppEngine object; 1999013 = Area object; 1999011 = DDESuiteLinkClient, OPCClient, and InTouchProxy objects; 1999024 = RedundantDIObject object; 1999033 = Undeployed object represented by a generic name; 1999901 = ApplicationObject; 1999902 = Traceability object.

Column	Data Type	Description
Name	nvarchar(255) NULL	The name of this object in the hierarchy.
ConfigStor	ntext(16) NULL	If the namespace object has configuration information associated with it (for example, configuration information for a set of trend curves, the name of the file that contains the configuration information).
(FK) ParentKey	int NOT NULL	The unique identifier for a named object in this namespace.

PublicGroupTag

Contains one row for each instance of a tag in the public namespace.

Column	Data Type	Description
(FK) NameKey	int NOT NULL	The unique identifier for the object in the namespace. NameKey is a foreign key from the PublicNameSpace table.
(FK) wwDomainTagKey	int NOT NULL	The unique numerical identifier for a tag in a specific domain. wwDomainTagKey is a foreign key from the TagRef table.

PublicNameSpace

Contains one row for each object in the public namespace. Objects in the public namespace can include plant machines, areas, and so on, and are organized in a hierarchy. Allows more than one name to map to a single tag.

Column	Data Type	Description
NameKey	int IDENTITY	The unique identifier for the object in the namespace. This value is automatically generated by the system when the object is added.

Column	Data Type	Description
Type	int NULL	The value that specifies the type of namespace. 1 to 6 = Tag; 1 to 2 million = System; 2+ million = Groups. Within the system range, the following values designate ArchestrA object types: 1999023 = Galaxy; 1999001 = WinPlatform object; 1999003 = AppEngine object; 1999013 = Area object; 1999011 = DDESuiteLinkClient, OPCClient, and InTouchProxy objects; 1999024 = RedundantDIObject object; 1999033 = Undeployed object represented by a generic name; 1999901 = ApplicationObject; 1999902 = Traceability object.
Name	nvarchar(255) NULL	The name of this object in the hierarchy.
ConfigStor	ntext NULL	If the namespace object has configuration information associated with it (for example, configuration information for a set of trend curves, the name of the file that contains the configuration information).
(FK) ParentKey	int NOT NULL	The unique identifier for a named object in this namespace.
OriginalName	nvarchar(255) NOT NULL	Internal use only.

QualityMap

Contains one row for every permutation of quality detail for a tag value.

Column	Data Type	Description
QualityDetail	int NOT NULL	An internal representation of data quality.
QualityString	nvarchar(max) NULL	The text string that describes what the quality detail value means.

RateOfChange

Contains one row for each monitored rate of change for a tag.

Column	Data Type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
(FK) ContextKey	int NOT NULL	The unique numerical identifier for the context. ContextKey is a foreign key from the Context table.
Value	float(8) NOT NULL	The percentage of change for a tag during the amount of time specified by the time base.
TimeBase	int NOT NULL	The unit of time against which the rate of change will be measured.
Priority	int NOT NULL	The priority for the rate of change. Priorities can range from 1 to over 2 billion, with 1 being the highest priority.
Checked	bit NOT NULL	Used to specify whether a tag imported from InTouch was configured for automatic rate of change checking. 0 = Checking disabled; 1 = Checking enabled.

ReplicationGroup

Contains one row for each replication group. (This is used exclusively for tiered historian installations.)

Column	Data Type	Description
ReplicationGroupKey	int NOT NULL	The unique identifier for the replication group.
ReplicationGroupName	nvarchar(255) NOT NULL	The name of the replication group.
(FK) ReplicationServerKey	int NOT NULL	The unique identifier for the replication server. ReplicationServerKey is a foreign key from the ReplicationServer table.

Column	Data Type	Description
(FK) ReplicationTypeKey	tinyint NOT NULL	Can be 1, 2, or 3. (1 = Simple Replication, 2 = Analog Summary Replication, 3 = State Summary Replication.) ReplicationTypeKey is a foreign key from the ReplicationType table.
(FK) ReplicationScheduleKey	int NOT NULL	The unique identifier for the replication schedule. ReplicationScheduleKey is a foreign key from the ReplicationSchedule table.
SummaryReplicationNamingScheme	nvarchar(512) NULL	The naming scheme for the replication tags belonging to this replication group. If the summary replication naming scheme is NULL, the summary replication naming scheme from the replication server is used as the default naming scheme for summary tags.
GroupAbbreviation	nvarchar(32) NULL	The abbreviation for the replication group. If GroupAbbreviation is NULL, ScheduleAbbreviation is used as the default group abbreviation.
Status	tinyint NULL	Automatically updated by the system if a change is made to the replication group: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.

ReplicationSchedule

Contains one row for each replication schedule. (This is used exclusively for tiered historian installations.)

Column	Data Type	Description
ReplicationScheduleKey	int NOT NULL	The unique identifier for the schedule.
ReplicationScheduleName	nvarchar(255) NOT NULL	The name of the replication schedule.
(FK) ReplicationScheduleType	int NOT NULL	The type of replication schedule. ReplicationScheduleType is a foreign key fom the ReplicationScheduleType table.
ReplicationScheduleAbbreviation	nvarchar(32) NOT NULL	The abbreviation for the replication schedule.
CreateGroup	bit NOT NULL	If TRUE, this replication schedule is automatically added to new replication groups.

ReplicationScheduleType

Contains one row for each type of replication schedule. (This is used exclusively for tiered historian installations.)

Column	Data Type	Description
ReplicationScheduleTypeKey	int NOT NULL	The unique identifier for the schedule type.
ReplicationScheduleTypeName	nvarchar(32) NOT NULL	The name of the replication schedule type, either INTERVAL or CUSTOM. The default is INTERVAL.

ReplicationServer

Contains one row for each replication server. (This is used exclusively for tiered historian installations.) The password is encrypted by an internal routine before storing in this table.

Column	Data Type	Description
ReplicationServerKey	int NOT NULL	The unique identifier for the replication server.
ReplicationServerName	nvarchar(255) NOT NULL	The name of the replication server.
Description	nvarchar(512) NULL	The description of the replication server.
SFPath	nvarchar(260) NULL	The local store-and-forward path associated with the replication server for this instance of Wonderware Historian.
SFFreeSpace	int NOT NULL	The free space for the store-and-forward path in MB.
AuthenticateWithAAUser	bit NULL	1 if the login should be authenticated using the ArchestrA user name; otherwise, 0 to authenticate with the UserName and Password.
UserName	nvarchar(255) NULL	The user name for logging in to the replication server. (AuthenticateWithAAUser must be 0.)
Password	nvarchar(512) NULL	The encrypted password for logging in to the replication server. (AuthenticateWithAAUser must be 0.)
TCPPort	int NOT NULL	The TCP port to use to log in to the replication server.

Column	Data Type	Description
SummaryReplicationNamingScheme	nvarchar(512) NULL	The naming rule for summary replication tags. If ReplicationGroupKey is NULL, the naming rule is used from the ReplicationServerName scheme. If ReplicationServerName is NULL, the naming rule is used from the SummaryReplicationNamingScheme system parameter.
SimpleReplicationNamingScheme	nvarchar(512) NULL	Naming rule for simple replication tags. If NULL the naming rule specified in the simple replication naming scheme system parameter is used.
BufferCount	int NOT NULL	The number of 64 KB buffers pre-allocated for buffering data. This number may need to be increased to accommodate high data rates. This value is of data type int, with a default of 128.
Bandwidth	int NOT NULL	The bandwidth in kbps used between tier-1 and tier-2. -1 = unlimited.

Column	Data Type	Description
MinSFDuration	int NOT NULL	The minimum duration, in seconds, for the replication service server node to function in store-and-forward mode. The replication service server node functions in store-and-forward mode for this length of time even if the condition that caused replication service server node to function in store-and-forward mode no longer exists. The maximum duration is 3600 seconds, and the minimum is 0 seconds.
Status	tinyint	Automatically updated by the system if a change is made to the replication server: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.

ReplicationSyncRequest

Contains one row for each replication synchronization request. (This is used exclusively for tiered historian installations.)

Column	Data Type	Description
ReplicationSyncRequestKey	bigint NOT NULL	The unique identifier for the replication synchronization request.
ReplicationTagEntityKey	int NOT NULL	The unique identifier for the replication tag entity.
RequestVersion	smallint NOT NULL	The version type. 0 = Initial version; 1 = Latest version.
ModStartDateTimeUtc	datetime NOT NULL	The start time (in UTC) for the replication synchronization request.

Column	Data Type	Description
ModStopDateTimeUtc	datetime NOT NULL	The stop time (in UTC) for the replication synchronization request.
EarliestExecutionDateTimeUtc	datetime NULL	The ealiest execution date (in UTC) for the replication synchronization request.
ExecuteState	tinyint NOT NULL	Value automatically changes as the rep service processes the sync queue. 0 = ready to process; 1 = currently being processed; 2 = rows needs merging/unmerging.

ReplicationTag

Contains one row for each replication tag. (This is used exclusively for tiered historian installations.) Replication tags follow the same naming convention as regular tags.

Column	Data Type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
SourceTag	nvarchar(256) NOT NULL	The name of the source tag used for the replication tag.
SourceServer	nvarchar(255) NOT NULL	The name of the tier 1 server with the source tag.

ReplicationTagEntity

Contains one row for each replication tag entity. (This is used exclusively for tiered historian installations.)

Column	Data Type	Description
ReplicationTagEntityKey	int NOT NULL	The unique identifier for the replication tag entity.

Column	Data Type	Description
(FK) ReplicationServerKey	int NOT NULL	The unique identifier for the replication server. ReplicationServerKey is a foreign key from the Replication Server table.
DestinationTagName	nvarchar(256) NOT NULL	The name of the destination tag. If the destination tag name is not specified, it is generated based on the naming convention for the replication tag and stored in the database.
DestinationTagID	uniqueidentifier NOT NULL	The unique identifier for the destination tag.
(FK) SourceTagName	nvarchar(256) NOT NULL	The name of the source tag. SourceTagName is a foreign key from the Tag table.
(FK) ReplicationGroupKey	int NOT NULL	The unique identification for the replication group. ReplicationGroupKey is a foreign key from the Replication Group table.
MaximumStates	tinyint NOT NULL	Maximum number of states to track for state summary tags. Discrete summary tags have a limit of 3 states. Analog summary tags of a limit of 100 states. The default is 10 states.

Column	Data Type	Description
(FK) CurrentEditor	tinyint NOT NULL	Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server.
Status	tinyint NULL	Automatically updated by the system if a change is made to the replication tag: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.

ReplicationType

Contains one row for each replication type. (This is used exclusively for tiered historian installations.)

Column	Data Type	Description
ReplicationTypeKey	tinyint NOT NULL	Can be 1, 2, or 3.
ReplicationTypeName	nvarchar(255) NOT NULL	Value is determined by the ReplicationTypeKey. 1 = Simple Replication, 2 = Analogy Summary Replication, 3 = State Summary Replication.

ServerList

Contains one row for each server used in an enterprise system. Allows for the creation of the system namespace, which contains a list of servers, and a flat namespace of tags per server.

Column	Data Type	Description
ServerKey	int IDENTITY	The unique numerical identifier of a Wonderware Historian server. This value is automatically generated by the system when a server is added.
ComputerName	nvarchar(50) NOT NULL	The Microsoft network name of the server computer.
Description	nvarchar(50) NULL	The description of the server.

SnapshotDetail

Contains one row for each storage size configuration for tags. This table is used by the storage subsystem to manage the snapshot files.

Column	Data Type	Description
StorageSize	int NOT NULL	The storage size, in bytes, of the tag value: -1 = Blob; 0 = Variable length string; 1 = 1 byte; 2 = 2 byte; 4 = 4 byte; 8 = 8 byte.

Column	Data Type	Description
SnapshotSize	int NOT NULL	The maximum size of the snapshot, in bytes. If this limit is reached, a new snapshot is created. The default is 2,097,152 bytes (2 MB).
ImageTime	int NOT NULL	The interval, in seconds, between updates to the snapshot file. The snapshot file is updated with tag value information from the snapshot buffer, which resides in memory. The default is 30 seconds, and the maximum value is 60 seconds.
ThresholdTime	int NOT NULL	The maximum amount of time, in seconds, that can elapse before a new snapshot is automatically created, provided that the value for the snapshot size has not been reached. The default is 3600 seconds (1 hour).
Status	tinyint NULL	Automatically updated by the system if a change is made to the snapshot: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.

SnapshotTag

Contains one row for each tag that is included in the snapshot action associated with a given event tag.

Column	Data Type	Description
SnapshotTagKey	int IDENTITY	The unique numerical identifier of the tag included in the snapshot. This value is automatically generated by the system when the snapshot is added.
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. This tag is the snapshot tag. TagName is a foreign key from the Tag table.
(FK) EventTagName	nvarchar(256) NOT NULL	The name of the event tag to which the snapshot tag is related. EventTagName is a foreign key from the EventTag table.
(FK) TagType	int NOT NULL	Used to indicate the type of tag. 1 = Analog; 2 = Discrete; 3 = String. The default is 1. TagType is a foreign key from the TagRef table.

SQLTemplate

Contains one row for each pre-defined SQL script, which can be copied and used as a basis for an event detection or action script.

Column	Data Type	Description
TemplateKey	int IDENTITY	The unique numerical identifier for a SQL template. This value is automatically generated when a new SQL template is created.
Description	nvarchar(50) NULL	The description of the SQL template.
Type	int NULL	The type of SQL template. 0 = Detector; 1 = Action.
Script	ntext(16) NULL	A pre-defined SQL script. This script can be copied and used as an event detection or action script.

StateSummaryHistory (INSQL.Runtime.dbo.StateSummaryHistory)

The StateSummaryHistory extension table returns results for state summary points.

Column	Data Type	Description
TagName	nvarchar(256) NOT NULL	The tag name.
StartDateTime	datetime / datetime2 (7) NOT NULL	Start time of retrieval cycle. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
EndDateTime	datetime / datetime2 (7) NOT NULL	End time of retrieval cycle. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
Value	float(53) NULL	Numeric state.

Column	Data Type	Description
vValue	sql_variant NULL	Non-numeric state.
OPCQuality	int NULL	OPC quality. Normal OPC quality retrieval logic is applied if: • All the point found and processed for this row have GOOD quality. If they all have the same GOOD quality, then that quality is returned. • If there is a gap in the entire calculation cycle, then BAD quality is returned for the tag. • For any other scenario with any mixture of GOOD and BAD points, a DOUBTFUL OPC quality (64) is returned. For more information, see "Quality Rule (wwQualityRule)" in Chapter 7, "Data Retrieval Options," in your *Wonderware Historian Concepts Guide*.
StateCount	int NULL	The number of times the state occurred within the retrieval cycle, including states that only partially occur in the cycle.
ContainedStateCount	int NULL	The number of times the state occurred fully contained within the retrieval cycle. States that only partially occur in the cycle are not counted.
StateTimeMin	float(53) NULL	Minimum time in this state among all occurrences of this state during this retrieval cycle, including state occurrences that fall only partially within the period. An occurrence that was partially contained in two or more consecutive storage cycles is converted to a contained state within the retrieval cycle if possible.

Column	Data Type	Description
StateTimeMinContained	float(53) NULL	The minimum of the contained times in this state among all occurrences of this state during the entire retrieval cycle, excluding state occurrences that fall only partially within the period. An occurrence that was partially contained in two or more consecutive storage cycles is converted to a contained state within the retrieval cycle if possible.
StateTimeMax	float(53) NULL	Maximum time in this state among all occurrences of this state during this retrieval cycle, including state occurrences that fall only partially within the period. An occurrence that was partially contained in two or more consecutive storage cycles is converted to a contained state within the retrieval cycle if possible.
StateTimeMaxContained	float(53) NULL	The maximum of the contained times in this state among all occurrences of this state during the entire retrieval cycle, excluding state occurrences that fall only partially within the period. An occurrence that was partially contained in two or more consecutive storage cycles is converted to a contained state within the retrieval cycle if possible.
StateTimeAvg	float(53) NULL	Average time in this state among all occurrences of this state during this retrieval cycle, including state occurrences that fall only partially within the period.

Column	Data Type	Description
StateTimeAvgContained	float(53) NULL	Average time in this state among all occurrences of this state during this retrieval cycle, excluding state occurrences that fall only partially within the period. An occurrence that was partially contained in two or more consecutive storage cycles is converted to a contained state within the retrieval cycle if possible.
StateTimeTotal	float(53) NULL	Total time in this state during this retrieval cycle, including state occurrences that fall only partially within the period.
StateTimeTotalContained	float(53) NULL	Total time in this state during this retrieval cycle, excluding state occurrences that fall only partially within the period. An occurrence that was partially contained in two or more consecutive storage cycles is converted to a contained state within the retrieval cycle if possible.
StateTimePercent	float(53) NULL	Percent of the time during this retrieval cycle that the tag was in this state, including state occurrences that fall only partially within the period.
StateTimePercentContained	float(53) NULL	The precentage of the entire retrieval cycle time that the tag was in this state, excluding state occurrences that fall only partially within the period. This is a ratio between StateTimeTotalContained and StateTimeTotal expressed as a percentage in the range 0 to 100. An occurrence that was partially contained in two or more consecutive storage cycles is converted to a contained state within the retrieval cycle if possible.
SourceTag	nvarchar(256) NULL	The source (tier 1) tag for the summary tag.
SourceServer	nvarchar(256) NULL	The source (tier 1) server for the summary tag.

Column	Data Type	Description
wwCycleCount	int NULL	The number of cycles into which the entire query time range has been divided.
wwResolution	int NULL	Length of cycles in milliseconds. The default is 3600000 (equal to 1 hour).
wwTimeZone	nvarchar(50) NULL	Time zone to use for interpreting both input and output timestamp parameters. If none is specified, then the default is set to LOCAL.
wwVersion	nvarchar(30) NULL	Data version, ORIGINAL or LATEST. If none is specified, the default is LATEST.
wwTagKey	int NOT NULL	Tag key.
wwRetrievalMode	nvarchar(16) NULL	Determines whether to use CYCLIC or DELTA retrieval. The default is DELTA.
wwMaxStates	int NULL	The maximum number of states (for state summaries) that are stored. The first N states will have summary values. For internal use only.

StateWideHistory (INSQL.Runtime.dbo.StateWideHistory)

Contains one row for the amount of time one or more analog, discrete, or string tags have been in a particular state, thus providing a "wide" view of the data.

Column	Data Type	Description
DateTime	datetime / datetime2 (7) NOT NULL	The timestamp for the start of the time-in-state period. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)

Column	Data Type	Description
vValue	sql_variant NULL	The string representation of the state, the ordinal for state types that do not have a string representation, or NULL for a gap or "bad" value.
Tag1	float(25) NULL	The name of a tag to query.
Tag2	float(25) NULL	The name of a tag to query.
ManyOtherTags	float(25) NULL	A "placeholder" column for one or more tags in the wide table format. In the wide table format, tagnames are used as column names. The ManyOtherTags column is "duplicated" for as many tags as are specified in the database query.
wwRowCount	int NULL	The number of rows to be returned for a specified time period. For cyclic retrieval, the rows are spaced evenly across the time period, and the default row count is 100 rows. For cyclic retrieval, the row count is applied for each tag in a query. This parameter has been deprecated; do not use. Use the wwCycleCount parameter instead.
wwResolution	int NULL	The sampling rate, in milliseconds, for retrieving the data in cyclic mode. The system returns values stored over the requested time period at the interval specified by the resolution. For example, if you specify a 5000 ms resolution, the system queries for all data during the time period and then only returns those values that occur at each 5000 ms interval, starting with the start date and ending with the end date.
wwRetrievalMode	nvarchar(16) NULL	Used to specify the time-in-state retrieval mode. The valid values are VALUESTATE and ROUNDTRIP. The default wwRetrievalMode is VALUESTATE.
wwTimeDeadband	int NULL	The minimum time, in milliseconds, between returned values for a single tag. Applies only to delta retrieval.

Column	Data Type	Description
wwValueDeadband	real NULL	The percentage of full scale (range), in engineering units. Any value changes that are less than this percentage are not returned. Applies only to delta retrieval. The default is 0.
wwTimeZone	nvarchar(50) NULL	Used to specify the time zone for retrieval. By default, the retrieval subsystem converts the UTC (Universal Time Coordinated) timestamps for the stored data to the local time of the Wonderware Historian computer, including adjustments for daylight savings time. To keep the timestamps in UTC, set the value of wwTimeZone to UTC. To convert the timestamps to a client computer's time zone, set this parameter to the appropriate time zone text key from the TimeZone table.
wwVersion	nvarchar(30) NULL	If the original data values have been modified in the database, use this column to specify which version of the stored data is to be retrieved. Valid values are: ORIGINAL or LATEST. If no parameter is specified, the latest version of the data is retrieved by default. Modification is indicated by the QualityDetail.
wwCycleCount	int NULL	The number of retrieval cycles (sub-intervals) for the specified time period. The cycles will be spaced evenly across the time period. For example, if you specify a cycle count of four, the time period will be divided into four even cycles, and one or more values (depending on the retrieval mode) will be returned per cycle.
wwTimeStampRule	nvarchar(20) NULL	Used to specify whether cyclic results are timestamped at the beginning of the cycle or at the end of the cycle. Valid values are START and END. If no timestamp rule is specified in the query, then retrieval uses the setting of the TimeStampRule system parameter.

Column	Data Type	Description
wwQualityRule	nvarchar(20) NULL	Used to specify whether values with certain characteristics are explicitly excluded from consideration by data retrieval. This parameter will override the setting of the QualityRule system parameter. Valid values are GOOD, EXTENDED, or OPTIMISTIC. A quality rule of GOOD means that data values with doubtful (64) OPC quality will not be used in the retrieval calculations and will be ignored. Values with bad QualityDetail indicate gaps in the data. A quality rule of EXTENDED means that data values with both good and doubtful OPC quality will be used in the retrieval calculations. Values with bad QualityDetail indicate gaps in the data. A quality rule of OPTIMISTIC means that calculations that include some good and some NULL values will not cause the overall calculations to return NULL. You can apply wwQualityRule to all retrieval modes.
wwStateCalc	nvarchar(20) NULL	Used to indicate the type of calculation to return in the StateTime column for the "value state" retrieval mode. Valid values are: MINIMUM, MAXIMUM, AVERAGE, TOTAL, CONTAINED, or PERCENT. You can also use the shortened versions: MIN, MAX, AVG, or SUM. The default for this column is TOTAL.
wwParameters	nvarchar(128) NULL	Used for additional parameters that can be specified. By default, the value of this parameter is an empty string.
StartDateTime	datetime / datetime2 NOT NULL	Start time of the retrieval cycle for which this row is returned. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)

Column	Data Type	Description
wwFilter	nvarchar(512) NOT NULL	The name of the filter. Filters are specified as C-like functions and parentheses are always required, even when the filter does not override the default parameters (no parameters are passed). Filter values are NoFilter, ToDiscrete(), SigmaLimit(), and SnapTo(). The default value is NoFilter. If the query does not specify the wwFilter element at all, or if its default value is not overridden, then no filter is applied.
wwMaxStates	int NULL	For internal use only.

StorageLocation

Contains one row for each defined storage location on a specific storage node.

Column	Data Type	Description
StorageType	int NOT NULL	The type of storage used for the specified location. 1 = Circular; 2 = Alternate; 3 = Buffer; 4 = Permanent. There can be only one storage location of each type.
SortOrder	int NOT NULL	Applies only to the alternate area. If more than one location is defined, the sort order determines the order in which the alternate areas are used. Reserved for future use.
(FK) StorageNodeKey	int NOT NULL	The unique numerical identifier for the storage node. StorageNodeKey is a foreign key from the StorageNode table.

Column	Data Type	Description
Path	nvarchar(255) NOT NULL	The path to the storage location. The circular storage location must be a local drive on the server machine, and the path must be specified using normal drive letter notation (for example, c:\Historian\Data\Circular). For a tier-1 historian, the alternate, buffer, and permanent storage locations can be anywhere on the network. For a tier-2 historian, the buffer and permanent storage locations can be anywhere on the network, but the alternate storage location must be on a local drive. The ArchestrA service user must have full access to network locations. The locations must be specified using UNC notation. Mapped drives are not supported. If empty, the default <SystemDataPath>\Wonderware\Data\Circular is used.
MaxMBSize	int NOT NULL	The limit, in megabytes, for the amount of data to be stored to the specified location. The maximum size applies to circular and alternate storage only. If the maximum size is set to 0, all available space at the storage location is used.
MinMBThreshold	int NOT NULL	The minimum amount of disk space, in megabytes, at which the system attempts to start freeing up space. The threshold applies to circular and alternate storage only. Typically, you should multiply the size of the average history block (before any compression) by 1.5 to determine the minimum threshold.
MaxAgeThreshold	int NOT NULL	The age, in days, of data that will be deleted by system to free up disk space. The threshold applies to circular and alternate storage only. The minimum age is 2 days. A value of 0 indicates that no age threshold is applied.
Status	tinyint NULL	Automatically updated by the system if a change is made to the storage location: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.

StorageNode

Contains one row for each defined storage node.

Note Only one storage node is supported for this release.

To satisfy referential integrity constraints between the StorageNode table and the Tag table, at least one row must exist in the StorageNode table and references to the column StorageNodeKey in the Tag table must exist as rows in this table.

Column	Data Type	Description
StorageNodeKey	int IDENTITY	The unique numerical identifier for the storage node. This value is automatically generated by the system when the storage node is added.
ComputerName	nvarchar(255) NOT NULL	The Microsoft network name of the computer on which the storage node resides.
Description	nvarchar(50) NULL	The description that identifies the role of the storage computer.
DbStatus	int NOT NULL	For releases prior to 8.0, used to store the status of server reinitializations. 2 = Certain columns in the Tag, AnalogTag, DiscreteTag, StringTag, Topic, and IOServer tables were changed; 3 = Reinitialization needed; 4 = Commit phase of a database update is in progress; 0 = Reinitialization complete. A negative value indicates that an error was encountered during reinitialization. Internal use only.
DbModAcquisition	int NOT NULL	Used with DbStatus to indicate to the back end whether the acquisition subsystem needs to be restarted. 0 = Restart not needed; 1 = Restart needed. Currently not used.
DbError	varchar(255) NULL	The description of the error that occurs if a database commit fails. Currently not used.
DbModStorage	int NOT NULL	Used with DbStatus to indicate to the back end whether the storage subsystem needs to be restarted. 0 = Restart not needed; 1 = Restart needed. Currently not used.
DbModServer	int NOT NULL	Used with DbStatus to indicate to the back end whether the database server needs to be restarted. 0 = Restart not needed; 1 = Restart needed. Currently not used.

Column	Data Type	Description
DbModAll	int NOT NULL	Used with DbStatus to indicate to the back end whether all subsystems need to be restarted. 0 = Restart not needed; 1 = Restart needed. This value overrides the value in all subsystem columns. Currently not used.
DbRevision	int NOT NULL	Current revision number of the database. This value is automatically incremented if DbStatus = 2.

StringSnapshot

Contains one row for each string tag value that was configured to be stored when a defined event occurred. To view analog, discrete, and string snapshot values at the same time, use the v_SnapshotData view instead. For more information, see "v_SnapshotData" on page 129.

Column	Data Type	Description
(FK) SnapshotTagKey	int NOT NULL	The unique numerical identifier of the tag included in the snapshot. SnapshotTagKey is a foreign key from the SnapshotTag table.
(FK) EventLogKey	int NOT NULL	The unique numerical identifier of an event occurrence. EventLogKey is a foreign key from the EventHistory table.
Value	nvarchar(512) NULL	The value of the string tag at the event timestamp.
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.
QualityDetail	int NULL	An internal representation of data quality.

StringTag

Contains one row for each defined string tag. Configuration information specific to string tags is stored in this table, while general information for all tag types is stored in the Tag table.

Column	Data Type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
MaxLength	smallint NOT NULL	The maximum number of characters for the string. Valid values are: 8, 16, 24, 32, 48, 64, 128, 131, 256, 512.
DoubleByte	tinyint NOT NULL	Used to specify whether or not to store the string as a double-byte string. 0 = Not stored as double-byte; 1 = Stored as double-byte. The default is 0.

StructureAttributes

Contains one row for each attribute definition for the StructureType read-only table.

Column	Data Type	Description
(FK) StructureID	uniqueidentifier NOT NULL	The unique identifier for the structure. StructureID is a foreign key from the StructureType table.
AttributeName	nvarchar(255) NOT NULL	The name of the structure attribute.
(FK) AttributeTypeKey	int NOT NULL	The unique identifier for the structure attribute. AttributeTypeKey is a foreign key from the AttributeType table.
AttributeOrder	tinyint NOT NULL	The order of the attribute within the structure.

StructureTag

Contains one row for each summary tag.

Column	Data Type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique numerical identifier for a SQL template. TagName is a foreign key from the Tag table.
(FK) StructureID	uniqueidentifier NOT NULL	The unique identifier for the structure. StructureID is a foreign key from the StructureType table.

StructureType

Contains one row for each structure type read-only table.

Column	Data Type	Description
StructureID	uniqueidentifier NOT NULL	The unique identifier for the structure.
StructureTypeName	nvarchar(255) NOT NULL	The name of the structure type.
Description	nvarchar(512) NOT NULL	The description of the structure type.

SummaryData

Contains one row for each summarized value, or result, for a tag. This table is used by the event subsystem; it is not used by the replication subsystem. The Quality column contains the highest quality value of the raw data from which the result is calculated.

Column	Data Type	Description
(FK) LogKey	int NOT NULL	The unique numerical identifier of the summary's historical log. LogKey is a foreign key from the SummaryHistory table.
(FK) SumVarKey	int NOT NULL	The unique numerical identifier for a summarized tag. SumVarKey is a foreign key from the SummaryTagList table.
Value	float(8) NULL	The value of the summary.
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.

Column	Data Type	Description
Modified	int NOT NULL	Used to specify whether or not the data has been modified. This value is optional. 1 = Modified; 0 = Not modified.

SummaryHistory

Contains one row for each occurrence of a summary operation. This table is used by the event subsystem; it is not used by the replication subsystem. Rows are inserted even if the operation did not return data.

Column	Data Type	Description
LogKey	int IDENTITY	The unique numerical identifier of the summary's historical log. This value is automatically generated by the system when the record is added.
(FK) OperationKey	int NOT NULL	The unique numerical identifier for the summary operation. OperationKey is a foreign key from the SummaryOperation table.
SummaryDate	datetime/datetime2 NOT NULL	The date applicable to the results of the calculation. It is either the time of the beginning or end of the calculation period, as specified by the summary operation definition. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
SumDateTimeStamp	tinyint NULL	Duplication of the TimeStamp column of the SummaryOperation table at the SummaryDate. This column allows you to keep the original calculation timestamp setting performed, in case of a later modification of the summary operation definition.
SumDateCalcType	varchar(3) NULL	Duplication of the CalcType column of the SummaryOperation table at the SummaryDate. This column allows you to keep the original calculation type performed, in case of a later modification of the summary operation definition.

Column	Data Type	Description
SumDateDuration	real NULL	Duplication of the Duration column of the SummaryOperation table at the SummaryDate. This column allows you to keep the original calculation duration used in case of a later modification of the summary operation definition.
SumDateResolution	int NULL	Duplication of the Resolution column of the SummaryOperation table at the SummaryDate. This column allows you to keep the original calculation resolution used, in case of a later modification of the summary operation definition.
Status	tinyint NOT NULL	The flag indicating the status of the operation. 0 = Operation completed successfully; Not 0 = Operation is in progress or has failed. Reserved for future use.
OperationStart	datetime/datetime2 NULL	The timestamp when the calculation started for the operation. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
OperationEnd	datetime/datetime2 NULL	The timestamp when the calculation completed for the operation. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)

SummaryOperation

Contains one row for each defined summary operation that is associated with the event tag specified in the TagName column. This table is used by the event subsystem; it is not used by the replication subsystem.

Column	Data Type	Description
OperationKey	int IDENTITY	The unique numerical identifier for the summary operation. This value is automatically generated by the system when the operation is added.
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the EventTag table.
(FK) CalcType	char(3) NOT NULL	The type of calculation to be performed: SUM, MAX, MIN, or AVG. CalcType is a foreign key from the CalcType table.
Description	nvarchar(50) NULL	The description of the summary operation.
Duration	real NOT NULL	The period, in seconds, for which the calculation is performed.
Resolution	int NOT NULL	The sampling rate, in milliseconds, for retrieving the data in cyclic mode. The system returns values stored over the requested time period at the interval specified by the resolution. For example, if you specify a 5000 ms resolution, the system queries for all data during the time period and then only returns those values that occur at each 5000 ms interval, starting with the start date and ending with the end date.
TimeStamp	tinyint NOT NULL	The timestamp to use when storing the result of the calculation. The timestamp can be either the time when the calculation period starts or ends. 0 = Beginning of the calculation period; 1 = End of the calculation period.
Frequency	nvarchar(12) NULL	The name for the frequency.
SourceType	varchar(3) NULL	The type of summary, set to 'DYN' (for "dynamic" data). Used for backward compatibility with Industrial Workbook.

SummaryTagList

Contains one row for each combination of a summarized tag and a specific summary operation. This table is used by the event subsystem; it is not used by the replication subsystem. This table is a linking table that allows tags to be associated with a type of operation.

Column	Data Type	Description
SumVarKey	int IDENTITY	The unique numerical identifier for a summarized tag. This value is automatically generated by the system when the summarized tag is added.
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
(FK) OperationKey	int NOT NULL	The unique numerical identifier for the summary operation. OperationKey is a foreign key from the SummaryOperation table.
LowerLimit	float(8) NULL	The lower limit of validity for the tag's value. Values lower than this limit are not used in the calculation. By default, this value is set to -1000000000.
UpperLimit	float(8) NULL	The upper limit of validity for the tag's value. Values higher than this limit are not used in the calculation. By default, this value is set to 1000000000.
Description	nvarchar(50) NULL	The description of the summarized tag. This normally describes the result of the operation, although this description can be the same as that of the tag on which the operation is performed.

SystemParameter

Contains one row for each system parameter.

Column	Data Type	Description
Name	nvarchar(50) NOT NULL	The unique name for the system parameter.
Value	sql_variant NULL	The value of the system parameter.

Column	Data Type	Description
Editable	bit NOT NULL	Used to determine if the value of the named system parameter can be changed using the InSQL Console. 1 = Editable; 0 = Not editable.
Description	nvarchar(255) NULL	The description of the system parameter.
Status	tinyint NULL	Automatically updated by the system if a change is made to the named system parameter: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.

Tag

Contains one row for each tag in the system and includes the basic definition for the tag, such as the I/O Server that supplies the values.

Column	Data Type	Description
TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system.
(FK) IOServerKey	int NULL	The unique numerical identifier for the I/O Server. IOServerKey is a foreign key from the IOServer table.
(FK) StorageNodeKey	int NOT NULL	The unique numerical identifier for the storage node. StorageNodeKey is a foreign key from the StorageNode table.
wwTagKey	int IDENTITY	The unique numerical identifier of a tag within a single Wonderware Historian. This value is automatically generated by the system when the tag is added.
(FK) TopicKey	int NULL	The unique numerical identifier for the topic. TopicKey is a foreign key from the Topic table.
Description	nvarchar(512) NULL	The description of the tag.

Column	Data Type	Description
AcquisitionType	tinyint NOT NULL	The method by which the tag's value is acquired. If the tag value is acquired from an I/O Server, specify the name of the I/O Server, topic, and item. 0 = Not acquired; 1 = Acquired via an I/O Server; 2 = Acquired via MDAS or a manual update; 3 = System driver.
StorageType	smallint NOT NULL	The type of storage defined for the tag. 0 = Not stored; 1 = Cyclic; 2 = Delta; 3 = Forced storage; 17 = The storage type has been changed from cyclic to "not stored." 18 = The storage type has been changed from delta to "not stored." 19 = The storage type has been changed from "forced" to "not stored."
AcquisitionRate	int NULL	For polled tags of AcquisitionType 1, the poll rate in milliseconds. Reserved for future use.
StorageRate	int NOT NULL	The rate at which the tag is stored if the storage type is cyclic. The rate is in milliseconds.
ItemName	nvarchar(256) NULL	The address string of the tag.
(FK) TagType	int NOT NULL	The type of tag. 1 = Analog; 2 = Discrete; 3 = String; 5 = Event, 7 = Summary tag (analog or state). TagType is a foreign key from the TagRef table.
TimeDeadband	int NULL	The minimum time, in milliseconds, between stored values for a single tag. Any value changes that occur within the time deadband are not stored. The time deadband applies to delta storage only. A time deadband of 0 indicates that the system will store the value of the tag each time it changes.
DateCreated	datetime NOT NULL	The date that the tag was created. If not specified, this date will be automatically generated.

Column	Data Type	Description
CreatedBy	nvarchar(256) NOT NULL	The name of the user or application that created the tag. If not specified, this name will be automatically generated.
(FK) CurrentEditor	tinyint NULL	Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server.CurrentEditor is a foreign key from CurrentEditor table.
SamplesInActiveImage	int NULL	The number of samples that the active image holds for the tag. 0 indicates that the active image is using the default of 65 values. The higher the number of samples, the higher the load on memory resources.

Column	Data Type	Description
AIRetrievalMode	tinyint NULL	Used to specify the behavior of retrieval for data in active image. You can either retrieve from all acquired data values that are currently in the active image, or only the data values that are configured to be stored on disk. Data on disk may be a subset of that in the active image, depending on the storage rate for the tag.Valid values are: 0 = All of the values received into the active image will be included in the returned data (default); 1 = Only the values that will be moved into storage will be included in the returned data.
Status	tinyint NULL	Automatically updated by the system if a change is made to the tag: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.
CalculatedAISamples	int NULL	The number of values required in the active image to hold data for 1 min (+15%), as calculated by the system. This value is updated only if the AIAutoResize system parameter is set to 1 and the number of required samples is greater than 65. This value is written to the SamplesInActiveImage column of the Tag table at system startup.
ServerTimeStamp	bit NOT NULL	Used to indicate whether local timestamping by the Wonderware Historian is used. 0 = The IDAS timestamp is used; 1 = The Wonderware Historian time is used for the timestamp. If a fast-changing tag is configured to use server timestamping, the packet of data that is sent to the storage subsystem may contain multiple data values with the same timestamp, which may affect data calculations, such as for swinging door storage.

Column	Data Type	Description
DeadbandType	smallint NOT NULL	The type of delta storage deadband to be applied for the tag. This setting is only in effect if delta storage is configured for the tag. 1= Time and/or value deadband; 2 = Rate (swinging door) deadband.
CEVersion	tinyint NOT NULL	The version number used to track changes to the information in the Tag table. Any change to the data in a row will cause the version indicator to change. The Configuration Editor (and other client tools) can detect the changed version and reload the corresponding tag details. Changes to this column are not tracked by the modification tracking system.
AITag	bit NOT NULL	Internal use only. Tier-1 tags will always have a value of 1. Tier-2 tags will always have a value of 0 for this column.
TagId	uniqueidentifier NOT NULL	The unique identifier for the tag.

TagRef

Contains one row for each tag in the system. This table is used as a reference table for the Tag table, so that the TagName column is not propagated as the primary key of child tables.

Column	Data Type	Description
wwDomainTagKey	int IDENTITY	The unique numerical identifier for a tag in a specific domain. This value is automatically generated by the system when the tag is added.
wwTagKey	int NOT NULL	The unique numerical identifier of a tag within a single Wonderware Historian. wwTagKey is populated from the Tag table, but is not a foreign key.
(FK) ServerKey	int NOT NULL	The unique numerical identifier of a Wonderware Historian server. ServerKey is a foreign key from the ServerList table.

Column	Data Type	Description
(FK) TagName	nvarchar(256) NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.
TagType	int NOT NULL	The type of tag. 1 = Analog; 2 = Discrete; 3 = String; 4 = Complex; 5 = Event, 7 = Summary tag (analog or state).

TagType

Contains one row for each tag type.

Column	Data Type	Description
TagTypeKey	int NOT NULL	The unique identifier for the tag type.
TagTypeName	nvarchar(32) NOT NULL	The name of the tag type.

TimeDetectorDetail

Contains at least one row for each event tag associated with a time detector.

Column	Data Type	Description
TimeDetectorDetailKey	int IDENTITY	The unique numerical identifier for each time-based event tag. This value is automatically generated by the system when a time-based event tag is created.
(FK) FrequencyID	int NOT NULL	The unique numerical identifier for the frequency. Used to link a frequency with a time-based detector. 1= Hourly; 2 = Daily; 3 = Weekly; 4 = Monthly; 5 = Periodic; 6 = Other (Reserved for future use). FrequencyID is a foreign key from the Frequency table.
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.

Column	Data Type	Description
Periodicity	int NULL	The interval period in minutes between detector events. Only used for a periodic detection.
StartDateTime	datetime NULL	The timestamp from which the time detector starts. Only used for a periodic detection.
RunTimeDay	tinyint NULL	In the context of a weekly detector, RunTimeDay maps the week day number (0 = Sunday – 6 = Saturday). In the context of a monthly detector, RunTimeDay maps to the day of the month. Not used for periodic detections.
RunTimeHour	tinyint NULL	The hour of the day at which the time detector triggers. Not used for periodic detections.
RunTimeMin	tinyint NULL	The minute of the hour at which the time detector triggers. Not used for periodic detections.

TimeDetectorDetailPendingDelete

Contains one row for each time detector that is pending deletion. This table is used internally by the system during the deletion process.

The columns in this table are the same as in the TimeDetectorDetail table.

TimeZone

Contains one row for each time zone as defined by the Windows® operating system. This table is automatically populated by the system.

Column	Data Type	Description
TimeZoneID	smallint NULL	The unique numerical identifier for the time zone.
TimeZone	nvarchar(100) NULL	The name of the time zone.

Column	Data Type	Description
Description	nvarchar(100) NULL	The description of the time zone that includes the hour offset from UTC (GMT) and major cities or regions in the time zone.
Offset	smallint NOT NULL	The offset, in minutes, for daylight savings time, when in effect.
RegistryName	nvarchar(100) NULL	The Windows registry name of the time zone, which is always in English.

Topic

Contains one row for each topic to be read from an I/O Server.

Column	Data Type	Description
TopicKey	int IDENTITY	The unique numerical identifier for the topic. This value is automatically generated by the system when the topic is added.
(FK) IOServerKey	int NOT NULL	The unique numerical identifier for the I/O Server. IOServerKey is a foreign key from the IOServer table.
(FK) StorageNodeKey	int NOT NULL	The unique numerical identifier for the storage node. StorageNodeKey is a foreign key from the StorageNode table.
Name	nvarchar(180) NOT NULL	The name of the topic.
TimeOut	int NOT NULL	The time span, in milliseconds, in which a data point must be received on the topic. If no data point is received in this time span, the topic is considered "dead." The historian disconnects and then attempts to reconnect to the topic.
Status	tinyint NULL	Automatically updated by the system if a change is made to the topic: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.
LateData	bit NOT NULL	Used to enable acquisition of "late" data. 0 = Late data disabled; 1 = Late data enabled.

Column	Data Type	Description
IdleDuration	int NOT NULL	The amount of time, in seconds, before data is processed from the I/O Server. For example, if you set this value to 60 seconds, data from this I/O Server is cached and only processed by the storage engine after no more data has been received from the I/O Server for at least 60 seconds.
ProcessingInterval	int NOT NULL	The amount of time, in seconds, after which late data from the I/O Server is processed, regardless of the idle duration. If the nature of the data is such that the idle duration is never satisfied, the historian storage engine processes data from the topic at least one time every processing interval. The processing interval defaults to twice the idle duration and cannot be set to a value less than the idle duration.

TopicImportInfo

Contains one row for each topic definition imported from an InTouch node.

Column	Data Type	Description
(FK) NodeKey	int NOT NULL	The unique numerical identifier of the named InTouch node. This value is automatically generated by the system when the node is added.
DdeSourceKey	int NOT NULL	The unique identifier for the DDE source. Assigned by the Wonderware Historian system when data is imported.
SourceName	nchar(50) NOT NULL	The DDE Access Name from InTouch.
ApplicationName	nchar(50) NULL	The name of the InTouch application from which the topic definition is imported.
TopicName	nchar(50) NOT NULL	The name of the topic definition that is imported.
RequestInitialData	bit NOT NULL	Used to determine if the topic was configured to request initial data. See the InTouch documentation for more information. Internal use only.

Column	Data Type	Description
AlwaysAdvise	bit NOT NULL	Used to determine if the topic was configured to poll all items for data. See the InTouch documentation for more information. Internal use only.
DefaultStorageRate	int NOT NULL	The cyclic storage rate for the topic. Internal use only.
DefaultStorageType	int NOT NULL	The default storage type for the topic. Internal use only.
TimeDeadband	int NOT NULL	The minimum time, in milliseconds, between returned values for a single tag. Applies only to delta retrieval.
ValueDeadband	float(25) NOT NULL	Either the InTouch log deadband or the Wonderware Historian deadband, as specified by the DeadbandType column. Internal use only.
DeadbandType	tinyint NULL	The type of deadband used. Internal use only.
Import	bit NOT NULL	Used to indicate whether the topic has previously been imported from InTouch into Wonderware Historian Internal use only.
ProtocolType	tinyint NOT NULL	The protocol used by the Wonderware Historian server to communicate with the I/O Server. Internal use only.
IODriverKey	int NULL	The unique identifier for an IDAS.
RateDeadband	float(25) NOT NULL	The rate deadband that was specified during the InTouch topic import. Internal use only. This rate deadband is not used for swinging door storage. For more information on the rate deadband for swinging door storage, see "AnalogSummaryTag" on page 36.

UserDetail

Contains one row for each Wonderware Historian user. Used to store additional user information that is not stored in the Microsoft SQL Server Runtime.sysusers table. Applicable for both users and groups of users.

When Wonderware Historian is installed, a SQL job is created on the Microsoft SQL Server that automatically updates this table every hour. In order for this job to run, the SQL Server Agent must be running. For more information about jobs, see your Microsoft Online Books.

Column	Data Type	Description
UserKey	int NOT NULL	The unique numerical identifier for a database user as defined in the UserDetail table.
UserName	nvarchar(128) NOT NULL	The name of the database user.
AccessLevel	int NOT NULL	The security level for the user. 1 is the lowest level and 9999 is the highest. Used to limit access of certain users.
uid	int NOT NULL	The identifier for the user. This ID is referenced from the Microsoft SQL Server sysusers table.
gid	int NOT NULL	The identifier for the group in which a user belongs. This ID is referenced from the Microsoft SQL Server sysusers table.

WideHistory (INSQL.Runtime.dbo.WideHistory)

Contains one row of values for multiple analog, discrete, or string tags for a single timestamp, thus providing a "wide" view of the data.

Because tagnames are used as column names for the returned data (indicated by Tag1, Tag2, and ManyOtherTags), the value data types will be of the appropriate type for associated tags.

Column	Data Type	Description
DateTime	datetime / datetime2 NOT NULL	The timestamp for the returned value. For delta retrieval, this is typically the time at which the value was acquired by the Wonderware Historian. For cyclic retrieval, this is the specific time requested or calculated (using a SQL function). (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)

Column	Data Type	Description
Tag1	(as per the tag type) NULL	The name of a tag to query.
Tag2	(as per the tag type) NULL	The name of a tag to query.
ManyOtherTags	(as per the tag type) NULL	A "placeholder" column for one or more tags in the wide table format. In the wide table format, tagnames are used as column names. The ManyOtherTags column is "duplicated" for as many tags as are specified in the database query.
wwRowCount	int NULL	The number of rows to be returned for a specified time period. For cyclic retrieval, the rows are spaced evenly across the time period, and the default row count is 100 rows. For cyclic retrieval, the row count is applied for each tag in a query. This parameter has been deprecated; do not use. Use the wwCycleCount parameter instead.
wwResolution	int NULL	The sampling rate, in milliseconds, for retrieving the data in cyclic mode. The system returns values stored over the requested time period at the interval specified by the resolution. For example, if you specify a 5000 ms resolution, the system queries for all data during the time period and then only returns those values that occur at each 5000 ms interval, starting with the start date and ending with the end date.
wwEdgeDetection	nvarchar(16) NULL	The type of edge detection result set that the query will return. Valid values are NONE, LEADING, TRAILING, and BOTH.

Column	Data Type	Description
wwRetrievalMode	nvarchar(16) NULL	Used to specify how retrieved data is processed before it is returned to the client. Valid values are: CYCLIC, DELTA, FULL, INTERPOLATED, BESTFIT, AVERAGE, MINIMUM, MAXIMUM, INTEGRAL, SLOPE, COUNTER, VALUESTATE, and ROUNDTRIP. FULL = All stored values are returned; CYCLIC = All stored data for tags during the specified time interval are returned for the number of retrieval cycles or resolution specified; DELTA = Only values that changed during the specified time interval are returned. For all other modes, a calculation is performed by the system on the data and the value(s) are returned. The default is CYCLIC for retrieval from analog tables, DELTA for retrieval from discrete and string tables, and default is DELTA for retrieval from the History table, unless the specific retrieval mode implies otherwise. For example, SLOPE always has DELTA characteristics. The default value for wwRetrievalMode is DELTA.
wwTimeDeadband	int NULL	The minimum time, in milliseconds, between returned values for a single tag. Applies only to delta retrieval.
wwValueDeadband	real NULL	The percentage of full scale (range), in engineering units. Any value changes that are less than this percentage are not returned. Applies only to delta retrieval. The default is 0.

Column	Data Type	Description
wwTimeZone	nvarchar(50) NULL	Used to specify the time zone for retrieval. By default, the retrieval subsystem converts the UTC (Universal Time Coordinated) timestamps for the stored data to the local time of the Wonderware Historian computer, including adjustments for daylight savings time. To keep the timestamps in UTC, set the value of wwTimeZone to UTC. To convert the timestamps to a client computer's time zone, set this parameter to the appropriate time zone text key from the TimeZone table.
wwVersion	nvarchar(30) NULL	If the original data values have been modified in the database, use this column to specify which version of the stored data is to be retrieved. Valid values are: ORIGINAL or LATEST. If no parameter is specified, the latest version of the data is retrieved by default. Modification is indicated by the QualityDetail.
wwCycleCount	int NULL	The number of retrieval cycles (sub-intervals) for the specified time period. The cycles will be spaced evenly across the time period. For example, if you specify a cycle count of four, the time period will be divided into four even cycles, and one or more values (depending on the retrieval mode) will be returned per cycle.
wwTimeStampRule	nvarchar(20) NULL	Used to specify whether cyclic results are timestamped at the beginning of the cycle or at the end of the cycle. Valid values are START and END. If no timestamp rule is specified in the query, then retrieval uses the setting of the TimeStampRule system parameter.

Column	Data Type	Description
wwInterpolationType	nvarchar(20) NULL	Used to determine which analog value to return at a given cycle boundary. Valid values are STAIRSTEP and LINEAR. If STAIRSTEP is specified, no interpolation occurs. The last known point is returned with the given cycle time. If no valid value can be found, a NULL is returned. If LINEAR is specified, the system calculates a new value at the given cycle time by interpolating between the last known value prior to the cycle time and the first value after the cycle time.
wwQualityRule	nvarchar(20) NULL	Used to specify whether values with certain characteristics are explicitly excluded from consideration by data retrieval. This parameter will override the setting of the QualityRule system parameter. Valid values are GOOD, EXTENDED, or OPTIMISTIC. A quality rule of GOOD means that data values with doubtful (64) OPC quality will not be used in the retrieval calculations and will be ignored. Values with bad QualityDetail indicate gaps in the data. A quality rule of EXTENDED means that data values with both good and doubtful OPC quality will be used in the retrieval calculations. Values with bad QualityDetail indicate gaps in the data. A quality rule of OPTIMISTIC means that calculations that include some good and some NULL values will not cause the overall calculations to return NULL. You can apply wwQualityRule to all retrieval modes.

Column	Data Type	Description
wwValueSelector	nvarchar(20) NULL	Used to specify which column to return for specified analog summary tags in the four basic retrieval modes, DELTA, FULL, CYCLIC, and INTERPOLATED. The defined set of selectors are AUTO (the default in all modes if not overridden), MINIMUM or MIN, MAXIMUM or MAX, FIRST, LAST, AVERAGE or AVG, and INTEGRAL. The default AUTO setting returns the Last attribute in the Value column (which makes it accessible in the WideHistory table). You can only override the selector for the basic retrieval modes.
wwFilter	nvarchar(512) NOT NULL	The name of the filter. Filters are specified as C-like functions and parentheses are always required, even when the filter does not override the default parameters (no parameters are passed). The default value is NoFilter. If the query does not specify the wwFilter element at all, or if its default value is not overridden, then no filter is applied.
wwParameters	nvarchar(128) NULL	Used for additional parameters that can be specified. By default, the value of this parameter is an empty string.
StartDateTime	datetime / datetime2 NOT NULL	Start time of the retrieval cycle for which this row is returned. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)

Chapter 3

Views

A view is a logical way of looking at data from one or more tables in the database. A view is a "virtual" table that does not actually exist in the database. A view contains pointers to the actual tables in the database. Views can be used to include a subset of information stored in one or more tables, while leaving out other information. Views are part of normal SQL Server functionality.

To make it easier to query data from some of the Wonderware Historian tables, a number of views onto the data are provided. Queries are performed on these views as if they were normal physical tables.

History Table Views

Views have been created for the extension tables to make querying these tables easier. Normally, you must use the full reference for an extension table in the query, which is **linked_server.catalog.schema.objectname**. An extension table view allows you to simply use the view name instead, eliminating the need to provide the long reference.

All of the following views reflect the same table structure as the extension tables after which they are named.

This view:	References this extension table:
History	INSQL.Runtime.dbo.History
HistoryBlock	INSQL.Runtime.dbo.HistoryBlock
Live	INSQL.Runtime.dbo.Live

This view:	References this extension table:
AnalogSummaryHistory	INSQL.Runtime.dbo.AnalogSummaryHistory
StateSummaryHistory	INSQL.Runtime.dbo.StateSummaryHistory

v_EventSnapshot

Returns one row for each snapshot value for an analog and/or discrete tag (specified by the TagName column) associated with a particular snapshot event (specified by the Event column).

Column	Data type	Description
Event	nvarchar(256) NULL	The unique name of the tag within the Wonderware Historian system.
EventTime	datetime/datetime2 (7) NOT NULL	The timestamp reflecting when event history data was acquired. This is the time when an event actually occurred. This time reflects the time zone of the Wonderware Historian. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
DetectionTime	datetime/datetime2 (7) NOT NULL	The timestamp reflecting when the event was detected by the event system. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
Edge	nvarchar(8) NOT NULL	The "edge" for the event detection. For more information on edge detection, see "Edge Detection for Events (wwEdgeDetection)" in Chapter 7, "Data Retrieval Options," in your *Wonderware Historian Concepts Guide*.
TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system.

Column	Data type	Description
Value	float(8) NULL	The value of the tag at the time of the event occurrence. Measured in engineering units.
Quality	tinyint NULL	The basic data quality indicator associated with the data value.
QualityDetail	int NULL	An internal representation of data quality.

Note When an event is not linked to a snapshot action, the **TagName** column is set to '-' and the **Value**, **Quality**, and **QualityDetail** columns are set to NULL.

v_EventStringSnapshot

Returns one row for each snapshot value for a string tag (specified by the TagName column) associated with a particular snapshot event (specified by the Event column).

Column	data type	Description
Event	nvarchar(256) NULL	The unique name of the tag within the Wonderware Historian system.
EventTime	datetime/datetime2 (7) NOT NULL	The timestamp reflecting when event history data was acquired. This is the time when an event actually occurred. This time reflects the time zone of the Wonderware Historian. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
DetectionTime	datetime/datetime2 (7) NOT NULL	The timestamp reflecting when the event was detected by the event system. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
Edge	nvarchar(8) NOT NULL	The "edge" for the event detection.

Column	data type	Description
TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system.
Value	nvarchar(512) NULL	The value of the string tag at the event timestamp.
Quality	tinyint NULL	The basic data quality indicator associated with the data value.
QualityDetail	int NULL	An internal representation of data quality.

v_ModTracking

Returns one row for each database modification made. For more information, see "Viewing Database Modifications" in Chapter 9, "Viewing or Changing System-Wide Properties," in your *Wonderware Historian Administration Guide*.

Column	Data type	Description
DateTime	datetime NOT NULL	The timestamp of when the modification occurred.
Table	varchar(50) NULL	The name of the modified object.
Column	nvarchar(30) NOT NULL	The name of the modified column.
ModType	char(1) NOT NULL	The type of modification. U = Update; I = Insert; D = Delete; 1 = SQL insert; 2 = SQL original insert; 3 = SQL update; 4 = CSV insert; 5 = CSV original insert; 6 = CSV update; 7 = CSV multi-point update; 8 = CSV "fast load" insert.
RowKey	sql_variant NOT NULL	The key identifier for the column modified in the table. For example, TagName for the Tag table, Name for the Topic table, and so on.
NewValue	sql_variant NULL	The new value stored in the column, if the modification was to a configuration table. For modifications to history data, this column contains the total count of consecutive value updates attempted.

Column	Data type	Description
OldValue	sql_variant NULL	The value stored in the column before the modification was made, if the modification was to a configuration table. For modifications to history data using SQL INSERT and UPDATE statements, this column contains the timestamp of the earliest data affected by the INSERT or UPDATE operation. If multiple changes are made to the same data, then only the most recent change will be contained in this column. This column is not used for modifications made to history data using a CSV file.
User	nvarchar(256) NOT NULL	The name of the database user that made the modification. The value of this column reflects the Windows authentication user name (for example, DOMAIN\user_login_name) or the SQL Server authentication user name (for example, dbo), depending on how the user is logged into the SQL Server when the modification is made. In the case of a CSV file import, this column contains the user name as it appears in the CSV file.

v_SnapshotData

Returns one row for each snapshot value for an analog, discrete, and/or string tag (specified by the TagName column) associated with a particular snapshot event (specified by the Event column).

Column	Data type	Description
Event	nvarchar(256) NULL	The name of the event tag to which the snapshot tag is related.

Column	Data type	Description
EventTime	datetime/datetime2 (7) NOT NULL	The timestamp reflecting when event history data was acquired. This is the time when an event actually occurred. This time reflects the time zone of the Wonderware Historian. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
DetectionTime	datetime/datetime2 (7) NOT NULL	The timestamp reflecting when the event was detected by the event system. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
Edge	nvarchar(8) NOT NULL	The "edge" for the event detection.
TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system.
Value	nvarchar(512) NULL	The value of the snapshot tag at the event timestamp.
Quality	tinyint NULL	The basic data quality indicator associated with the data value.
QualityDetail	int NULL	An internal representation of data quality.

ReplicationSyncRequestInfo

Contains one row for each replication synchronization request. (This is used exclusively for tiered historian installations.)

Column	Data Type	Description
SourceTagName	nvarchar(256) NULL	The name of the source tag used for the replication tag.
ReplicationServerName	nvarchar(255) NULL	The name of the replication server.

Column	Data Type	Description
DestinationTagName	nvarchar(256) NULL	The name of the destination tag.
EarliestExecutionDateTimeUtc	datetime NULL	The ealiest execution date (in UTC) for the replication synchronization request.
ModStartDateTimeUtc	datetime NOT NULL	The start time (in UTC) for the replication synchronization request.
ModStopDateTimeUtc	datetime NOT NULL	The stop time (in UTC) for the replication synchronization request.
ReplicationSyncRequestKey	bigint NOT NULL	The unique identifier for the replication synchronization request.
ReplicationTagEntityKey	int NOT NULL	The unique identifier for the replication tag entity.
RequestVersion	smallint NOT NULL	The version type. 0 = Initial version; 1 = Latest version.
ExecuteState	tinyint NOT NULL	Value automatically changes as the rep service processes the sync queue. 0 = ready to process; 1 = currently being processed; 2 = rows needs merging/unmerging.

Column	Data Type	Description
CurrentEditor	tinyint NULL	Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server.
DestinationTagID	uniqueidentifier NOT NULL	The unique identifier for the destination tag.
MaximumStates	tinyint NULL	Maximum number of states to track for state summary tags.
ReplicationGroupKey	int NULL	The unique identifier for the replication group.

Column	Data Type	Description
ReplicationServerKey	int NULL	The unique identifier for the replication server.
Status	tinyint NULL	Automatically updated by the system if a change is made to the replication server: 0 = No change; 1 = Insertion; 2 = Modification; 3 = Deletion.
AuthenticateWithAAUser	bit NULL	1 if the login should be authenticated using the ArchestrA user name; otherwise, 0 to authenticate with the UserName and Password.
Bandwidth	int NULL	The bandwidth in kbps used between tier-1 and tier-2. -1 = unlimited.
BufferCount	int NULL	The number of buffers.
Description	nvarchar(512) NULL	The description of the replication server.
MinSFDuration	int NULL	The minimum duration, in seconds, for the replication service server node to function in store-and-forward mode. The replication service server node functions in store-and-forward mode for this length of time even if the condition that caused replication service server node to function in store-and-forward mode no longer exists. The maximum duration is 3600 seconds, and the minimum is 0 seconds.
Password	nvarchar(512) NULL	The password for logging in to the replication server. (AuthenticateWithAAUser must be 0.)
SFFreeSpace	int NULL	The free space for the store-and-forward path in MB.

Column	Data Type	Description
SFPath	nvarchar(260) NULL	The local store-and-forward path associated with the replication server for this instance of Wonderware Historian.
ServerDefaultSimpleReplicationNamingScheme	nvarchar(512) NULL	Naming rule for simple replication tags. If NULL the naming rule specified in the SimpleReplicationNamingScheme system parameters is used.
ServerDefaultSummaryReplicationNamingScheme	nvarchar(512) NULL	The default naming rule for summary replication tags. If NULL, the naming rule specified in the SummaryReplicationNamingScheme system parameter is used.
TCPPort	int NULL	The TCP port to use to log in to the replication server.
UserName	nvarchar(255) NULL	The user name for logging in to the replication server. (AuthenticateWithAAUser must be 0.)
GroupAbbreviation	nvarchar(32) NULL	The abbreviation for the replication group.
ReplicationGroupName	nvarchar(255) NULL	The unique identifier for the replication group.
ReplicationScheduleKey	int NULL	The unique identifier for the schedule.
ReplicationTypeKey	tinyint NULL	Can be 1, 2, or 3. (1 = Simple Replication, 2 = Analog Summary Replication, 3 = State Summary Replication.)
GroupDefaultSummaryReplicationNamingScheme	nvarchar(512) NULL	The group default naming rule for summary replication tags.

Millisecond Resolution Differences in SQL Server 2008

If you are using Wonderware Historian with SQL Server 2008, timestamps retrieved with VTQs will use the datetime2 data type with the millisecond resolution. If you are using an earlier version of SQL Server, any query results will be returned using the datetime data type with its inherhent lower resolution.

The following table shows the columns that are changed to datetime2 data type when used with SQL Server 2008.

Table or view	Column
AnalogSummary	EndDateTime
AnalogSummary	FirstDateTime
AnalogSummary	LastDateTime
AnalogSummary	MaxDateTime
AnalogSummary	MinDateTime
AnalogSummary	StartDateTime
DynDailyAvg	SummaryDate
DynDailySum	SummaryDate
DynHourlyAvg	SummaryDate
DynHourlyMax	SummaryDate
DynHourlyMin	SummaryDate
DynHourlySum	SummaryDate
EventHistory	DateTime
EventHistory	DetectDateTime
History	DateTime
History	StartDateTime
HistoryBlock	FromDate
HistoryBlock	ToDate
HistoryBlock	ArchiveDate
Live	DateTime
StateSummary	StartDateTime
StateSummary	EndDateTime

Table or view	Column
StateWideHistory	DateTime
SummaryHistory	SummaryDate
SummaryHistory	OperationStart
SummaryHistory	OperationEnd
v_EventSnapshot	DetectionTime
v_EventSnapshot	EventTime
v_EventStringSnapshot	DetectionTime
v_EventStringSnapshot	EventTime
v_History	DateTime
v_History	StartDateTime
v_HistoryBlock	FromDate
v_HistoryBlock	ToDate
v_HistoryBlock	ArchiveDate
v_Live	DateTime
v_SnapshotData	DetectionTime
v_SnapshotData	EventTime
v_SummaryData	SummaryDate
WideHistory	DateTime

Chapter 4

Stored Procedures

A stored procedure is a pre-compiled group of SQL statements. By using a stored procedure, you can execute a group of sequentially performed actions in one query. In addition to the stored procedures supplied with Microsoft SQL Server, there are three types of stored procedures designed to be used with Wonderware Historian: system stored procedures, user stored procedures, and extended stored procedures.

Stored Procedures

Some stored procedures are useful when performing database queries to return information about specific tags in the system. These stored procedures allow you to return information on a tag's definition or to narrow the scope of a query on a data storage table. You can use these stored procedures when querying the database using ad-hoc query tools, such as SQL Server Management Studio.

Other stored procedures are used to configure Wonderware Historian. System stored procedures are normally run during startup and during changes to the system configuration. These stored procedures are used mainly by the historian setup program, the event subsystem, the System Management Console, and client applications.

Note Stored procedures prefixed with "ww_" are provided only for backward compatibility. For more information, see "Renamed Stored Procedures" on page 293.

aaActionStringSelect

Selects the action string for a specified event tag.

Syntax

```
aaActionStringSelect TagName
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaAddAnalogSummaryTag

Add an analog summary tag.

Syntax

```
aaAddAnalogSummaryTag TagName, TagId, Description,
  SourceTag, SourceServer, SourceTagRawType,
  SourceTagIntegerSize, SourceTagSignedInteger,
  CreatedBy, DateCreated, AcquisitionType,
  StorageNodeKey, IOServerKey, TopicKey, StorageType,
  EngineeringUnit, IntegralDivisor, MinEU, MaxEU,
  MinRaw, MaxRaw, DeadbandType, TimeDeadband,
  CurrentEditor, wwTagKey
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

TagId
The unique tag ID of the tag within the Wonderware Historian system. The value is of data type uniqueidentifier, with a default of NULL.

Description
The description of the analog summary tag. This value is of data type nvarchar(512), with a default of an empty string.

SourceTag
The name of the source tag to create the analog summary tag from. This value is of data type nvarchar(256), with a default of an empty string.

SourceServer
The name of the source server for the source tag. This value is of data type nvarchar(256), with a default of an empty string.

SourceTagRawType
The numeric type for the raw value. 1 = Euro Float, an outdated data type (4 bytes); 2 = MS Float (4 bytes); 3 = Integer (2 or 4 bytes); 4 = MS Double (reserved for future use) (8 bytes). This value is of data type int, with a default of 3.

SourceTagIntegerSize
The bit size of the analog tag. 12 = 12-bit; 15 = 15-bit; 16 = 16-bit; 32 = 32-bit; 64 = 64-bit (reserved for future use). This value is of data type tinyint, with a default of 16.

SourceTagSignedInteger
Used to specify whether an integer is a signed number (positive or negative) or an unsigned number (positive only). 0 = Unsigned; 1 = Signed. This value is of data type bit, with a default of 0.

CreatedBy
The name of the user or application that created the tag. This value is of data type nvarchar(256), with a default of an empty string.

DateCreated
The date that the tag was created. This value is of data type datetime, with a default of NULL.

AcquisitionType
The method by which the tag's value is acquired. If the tag value is acquired from an I/O Server, specify the name of the I/O Server, topic, and item. 0 = Not acquired; 1 = Acquired via an I/O Server; 2 = Acquired via MDAS or a manual update; 3 = System driver. This value is of data type tinyint, with a default of 1.

StorageNodeKey
The unique numerical identifier for the storage node. This value is of data type int, with a default of 1.

IOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with a default of NULL.

TopicKey
The unique numerical identifier for the topic. This value is of data type int, with a default of NULL.

StorageType
The type of storage defined for the tag. 0 = Not stored; 1 = Cyclic; 2 = Delta; 3 = Forced storage; 17 = The storage type has been changed from cyclic to "not stored." 18 = The storage type has been changed from delta to "not stored." 19 = The storage type has been changed from "forced" to "not stored." This value is of data type smallint, with a default of 2.

EngineeringUnit
The unit of measure. Examples are mph, grams, and pounds. This value is of data type nvarchar(32), with a default of an empty string.

IntegralDivisor
The factor to be applied when integrating a rate with the units [EngUnits/TimeUnit] to a quantity with units [EngUnits]. This factor is called the integral divisor. The default value of 1 assumes a time unit of seconds and ensures that a rate of [Unit/second] is correctly integrated to [Unit]. For a time unit of minutes, set the integral divisor value to 60; for a unit of hours, set the integral divisor value to 3600. The integral divisor is applied similarly to rates or quantities that are not expressed in terms of a time unit. For example, to convert watts to watt-hours, the integral divisor is 1/3600. To convert watts to kilowatt-hours, the integral divisor is 1/3600000. This value is of data type float(25), with a default of 1.

MinEU
The minimum value of the tag, measured in engineering units. This value is of data type float(25), with a default of 0.

MaxEU
The maximum value of the tag, measured in engineering units. This value is of data type float(25), with a default of 100.

MinRaw
The minimum value of the raw acquired value. This value is of data type float(25), with a default of 0.

MaxRaw
The maximum value of the raw acquired value. This value is of data type float(25), with a default of 4095.

DeadbandType
The type of delta storage deadband to be applied for the tag. This setting is only in effect if delta storage is configured for the tag. 1= Time and/or value deadband; 2 = Rate (swinging door) deadband. This value is of data type smallint, with a default of 1.

TimeDeadband
The minimum time, in milliseconds, between stored values for a single tag. Any value changes that occur within the time deadband are not stored. The time deadband applies to delta storage only. A time deadband of 0 indicates that the system will store the value of the tag each time it changes. This value is of data type int, with a default of 0.

CurrentEditor
Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server. This value is of data type int, with a default of 0.

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaAddReplicationGroup

Add or modify replication groups.

Syntax

```
aaAddReplicationGroup ReplicationGroupName,
  ReplicationServerName, ReplicationTypeKey,
  ReplicationScheduleName,
  SummaryReplicationNamingScheme, GroupAbbreviation,
  ReplicationGroupKey
```

Arguments

ReplicationGroupName
The name of the replication group. This parameter has to be specified, else will return an error. This value is of data type nvarchar(255), with no default.

ReplicationServerName
The name of the replication server. This value is of data type nvarchar(255), with no default.

ReplicationTypeKey
The type of replication. Value values are:

1 - Simple Replication

2 - Analog Summary Replication

3 - State Summary Replication

This value is of data type tinyint, with a default of 3.

ReplicationScheduleName
The name of the schedule. This value is of data type nvarchar(255), with no default.

SummaryReplicationNamingScheme
The naming scheme for summary replication tags. If not specified, the one specified in the ReplicationServer will be used. This value is of data type nvarchar(512), with a default of NULL.

GroupAbbreviation
GroupAbbreviation is used as part of naming. If not specified, the one specified in the Schedule will be chosen as group abbreviation. This value is of data type nvarchar(32), with a default of NULL.

ReplicationGroupKey
Unique identifier for the replication group. If specified, this will overwrite the properties of the replication group. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaAddReplicationSchedule

Add or modify the schedules for replication.

Syntax

```
aaAddReplicationSchedule ReplicationScheduleName,
  ReplicationScheduleTypeName,
  ReplicationScheduleAbbreviation, CreateGroup, Period,
  Unit, TimesOfDay, ReplicationScheduleKey
```

Arguments

ReplicationScheduleName
The name of the schedule. This parameter is required. This value is of data type nvarchar(255), with no default.

ReplicationScheduleTypeName
The name of the schedule type. Can be either INTERVAL or CUSTOM. This value is of data type nvarchar(32), with a default of INTERVAL.

ReplicationScheduleAbbreviation
Will be used when creating groups as group abbreviation if not specified. This value is of data type nvarchar(32), with a default of the schedule abbreviation.

CreateGroup
If set to TRUE, groups will be created automatically when the replication server is created. This value is of data type bit, with a default of 1.

Period
The period value. This parameter is required when the schedule type is INTERVAL. This value is of data type smallint, with a default of 0.

Unit
The name of the unit. This parameter is required when the schedule type is INTERVAL. This value is of data type nvarchar(32), with a default of an empty string.

TimesOfDay
A semicolon-separated list of the times of day. This parameter is required when the schedule type is CUSTOM. This value is of data type nvarchar(max), with a default of an empty string.

ReplicationScheduleKey
The unique identifier for the schedule. If specified, this will overwrite the properties of the identified schedule. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaAddReplicationServer

Add or modify replication servers.

Syntax

```
aaAddReplicationServer ReplicationServerName,
  Description, SFPath, SFFreeSpace,
  AuthenticateWithAAUser, UserName, Password, TCPPort,
  SummaryReplicationNamingScheme,
  SimpleReplicationNamingScheme, BufferCount,
  Bandwidth, MinSFDuration, ReplicationServerKey
```

Arguments

ReplicationServerName
Name or IP address of the tier 2 server. This value is of data type nvarchar(255), with a default of an empty string.

Description
Description of the replication server. This value is of data type nvarchar(512), with a default of an empty string.

SFPath
Store forward path for the replication server. The default is an empty string. This value is of data type nvarchar(260), with a default of an empty string.

SFFreeSpace
Free space for the store forward path in MB. This value is of data type int, with a default of 125.

AuthenticateWithAAUser
Set to 1 to authenticate with ArchestrA user. This value is of data type bit, with a default of 1.

UserName
User name for authenticating with tier 2 server. This value is nvarchar(255), with a default of NULL.

Password
Password for authenticating with tier 2 server. This value is of data type nvarchar(512), with a default of NULL.

TCPPort
TCP Port for communicating with tier 2 server. This value is of data type int, with a default of 32568.

SummaryReplicationNamingScheme
Naming rule for summary replication tags. If this is NULL, the naming rule specified in system parameters will be used. This value is of data type nvarchar(512), with a default of <ReplicationDefaultPrefix>.<SourceTagName>.<TypeAbbreviation><GroupAbbreviation>.

SimpleReplicationNamingScheme
Naming rule for simple replication tags. If this is NULL, the naming rule specified in System parameters will be used. This value is of data type nvarchar(512), with a default of <ReplicationDefaultPrefix>.<SourceTagName>.

BufferCount
The number of 64 KB buffers pre-allocated for buffering data. This number may need to be increased to accommodate high data rates. This value is of data type int, with a default of 128.

Bandwidth
The bandwidth in kbps used between tier-1 and tier-2. -1 = unlimited.

MinSFDuration
The minimum duration, in seconds, for the replication service server node to function in store-and-forward mode. The replication service server node functions in store-and-forward mode for this length of time even if the condition that caused replication service server node to function in store-and-forward mode no longer exists. The maximum duration is 3600 seconds, and the minimum is 0 seconds. This value is of data type int, with a default of 180.

ReplicationServerKey
Unique identifier for the replication server. If specified, this will overwrite the properties of the server identified by the key. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaAddReplicationTagEntity

Add or modify a replication tag entity.

Syntax

```
aaAddReplicationTagEntity SourceTagName,
  ReplicationGroupName, ReplicationServerName,
  ReplicationTypeKey, MaximumStates, CurrentEditor,
  DestinationTagId, DestinationTagName
```

Arguments

SourceTagName
The source tag name. This value is of data type nvarchar(256), with no default.

ReplicationGroupName
The name of the replication group. If this is NULL, the replication type is set to simple replication. This value is nvarchar(255), with a default of NULL.

ReplicationServerName
The name of the replication server. This value is of data type nvarchar(255), with no default.

ReplicationTypeKey
The type of replication. Valid values are:

1 - Simple Replication

2 - Analog Summary Replication

3 - State Summary Replication

This value is of data type tinyint, with a default of 2.

MaximumStates
Maximum number of states to track for state summary tags. This value is of data type tinyint, with a default of 10 and a maximum of 100.

CurrentEditor

0 - Historian

2 - WAS

This value is of data type tinyint, with a default of 0.

DestinationTagID
Unique ID of the destination tag. If NULL, the destination tag name is generated based on the naming rule. This value is of data type nvarchar(256), with a default of NULL.

DestinationTagName
Name of the destination tag. If NULL, the destination tag name is generated based on the naming rule. This value is of data type nvarchar(256), with a default of NULL.

ReplicationTagEntityKey
The unique identifier for the replication tag entity. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaAddStateSummaryTag

Add or modify a state summary tag.

Syntax

```
aaStateSummaryTag TagName, TagId, Description,
  SourceTag, SourceServer, CreatedBy, DateCreated,
  AcquisitionType, StorageNodeKey, IOServerKey,
  TopicKey, StorageType, DeadbandType, TimeDeadband,
  CurrentEditor, wwTagKey
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

TagId
The unique tag ID of the tag within the Wonderware Historian system. The value is of data type uniqueidentifier, with a default of NULL.

Description
The description of the analog summary tag. This value is of data type nvarchar(512), with a default of an empty string.

SourceTag
The name of the source tag to create the analog summary tag from. This value is of data type nvarchar(256), with a default of an empty string.

SourceServer
The name of the source server for the source tag. This value is of data type nvarchar(256), with a default of an empty string.

CreatedBy
The name of the user or application that created the tag. This value is of data type nvarchar(256), with a default of an empty string.

DateCreated
The date that the tag was created. This value is of data type datetime, with a default of NULL.

AcquisitionType
The method by which the tag's value is acquired. If the tag value is acquired from an I/O Server, specify the name of the I/O Server, topic, and item. 0 = Not acquired; 1 = Acquired via an I/O Server; 2 = Acquired via MDAS or a manual update; 3 = System driver. This value is of data type tinyint, with a default of 1.

StorageNodeKey
The unique numerical identifier for the storage node. This value is of data type int, with a default of 1.

IOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with a default of NULL.

TopicKey
The unique numerical identifier for the topic. This value is of data type int, with a default of NULL.

StorageType
The type of storage defined for the tag. 0 = Not stored; 1 = Cyclic; 2 = Delta; 3 = Forced storage; 17 = The storage type has been changed from cyclic to "not stored." 18 = The storage type has been changed from delta to "not stored." 19 = The storage type has been changed from "forced" to "not stored." This value is of data type smallint, with a default of 2.

DeadbandType
The type of delta storage deadband to be applied for the tag. This setting is only in effect if delta storage is configured for the tag. 1= Time and/or value deadband; 2 = Rate (swinging door) deadband. This value is of data type smallint, with a default of 1.

TimeDeadband
The minimum time, in milliseconds, between stored values for a single tag. Any value changes that occur within the time deadband are not stored. The time deadband applies to delta storage only. A time deadband of 0 indicates that the system will store the value of the tag each time it changes. This value is of data type int, with a default of 0.

CurrentEditor
Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server. This value is of data type int, with a default of 0.

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaAddStructureTag

Add or modify a structure tag.

Syntax

```
aaStructureTag TagName, TagId, Description, SourceTag,
  SourceServer, CreatedBy, DateCreated, StructureId,
  AcquisitionType, StorageNodeKey, IOServerKey,
  TopicKey, StorageType, EngineeringUnit,
  IntegralDivisor, MinEU, MaxEU, MinRaw, MaxRaw,
  DeadbandType, TimeDeadband, CurrentEditor, wwTagKey
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

TagId
The unique tag ID of the tag within the Wonderware Historian system. The value is of data type uniqueidentifier, with a default of NULL.

Description
The description of the analog summary tag. This value is of data type nvarchar(512), with a default of an empty string.

SourceTag
The name of the source tag to create the analog summary tag from. This value is of data type nvarchar(256), with a default of an empty string.

SourceServer
The name of the source server for the source tag. This value is of data type nvarchar(256), with a default of an empty string.

CreatedBy
The name of the user or application that created the tag. This value is of data type nvarchar(256), with a default of an empty string.

DateCreated
The date that the tag was created. This value is of data type datetime, with a default of NULL.

StructureId
The ID for the structure. The value is of data type uniqueidentifier, with a default of NULL.

AcquisitionType
The method by which the tag's value is acquired. If the tag value is acquired from an I/O Server, specify the name of the I/O Server, topic, and item. 0 = Not acquired; 1 = Acquired via an I/O Server; 2 = Acquired via MDAS or a manual update; 3 = System driver. This value is of data type tinyint, with a default of 1.

StorageNodeKey
The unique numerical identifier for the storage node. This value is of data type int, with a default of 1.

IOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with a default of NULL.

TopicKey
The unique numerical identifier for the topic. This value is of data type int, with a default of NULL.

StorageType
The type of storage defined for the tag. 0 = Not stored; 1 = Cyclic; 2 = Delta; 3 = Forced storage; 17 = The storage type has been changed from cyclic to "not stored." 18 = The storage type has been changed from delta to "not stored." 19 = The storage type has been changed from "forced" to "not stored." This value is of data type smallint, with a default of 2.

EngineeringUnit
The unit of measure. Examples are mph, grams, and pounds. This value is of data type nvarchar(32), with a default of an empty string.

IntegralDivisor
The factor to be applied when integrating a rate with the units [EngUnits/TimeUnit] to a quantity with units [EngUnits]. This factor is called the integral divisor. The default value of 1 assumes a time unit of seconds and ensures that a rate of [Unit/second] is correctly integrated to [Unit]. For a time unit of minutes, set the integral divisor value to 60; for a unit of hours, set the integral divisor value to 3600. The integral divisor is applied similarly to rates or quantities that are not expressed in terms of a time unit. For example, to convert watts to watt-hours, the integral divisor is 1/3600. To convert watts to kilowatt-hours, the integral divisor is 1/3600000. This value is of data type float(25), with a default of 1.

MinEU
The minimum value of the tag, measured in engineering units. This value is of data type float(25), with a default of 0.

MaxEU
The maximum value of the tag, measured in engineering units. This value is of data type float(25), with a default of 100.

MinRaw
The minimum value of the raw acquired value. This value is of data type float(25), with a default of 0.

MaxRaw
The maximum value of the raw acquired value. This value is of data type float(25), with a default of 4095.

DeadbandType
The type of delta storage deadband to be applied for the tag. This setting is only in effect if delta storage is configured for the tag. 1= Time and/or value deadband; 2 = Rate (swinging door) deadband. This value is of data type smallint, with a default of 1.

TimeDeadband
The minimum time, in milliseconds, between stored values for a single tag. Any value changes that occur within the time deadband are not stored. The time deadband applies to delta storage only. A time deadband of 0 indicates that the system will store the value of the tag each time it changes. This value is of data type int, with a default of 0.

CurrentEditor
Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server. This value is of data type int, with a default of 0.

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaAddTag

Add or modify a tag.

Important This stored procedure is used internally by the system. Do not use this stored procedure to add a new tag.

Syntax

```
aaAddTag wTagName, wTopicKey, wIOServerKey,
  wStorageNodeKey, wDescription, wAcquisitionType,
  wStorageType, wAcquisitionRate, wStorageRate,
  wItemName, wTagType, wTimeDeadband, wCurrentEditor
```

Arguments

wTagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

wTopicKey
The unique numerical identifier for the topic. This value is of data type int, with a default of NULL.

wIOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with a default of NULL.

wStorageNodeKey
The unique numerical identifier for the storage node. This value is of data type int, with a default of 1.

wDescription
The description of the analog summary tag. This value is of data type nvarchar(512), with a default of an empty string.

wAcquisitionType
The method by which the tag's value is acquired. If the tag value is acquired from an I/O Server, specify the name of the I/O Server, topic, and item. 0 = Not acquired; 1 = Acquired via an I/O Server; 2 = Acquired via MDAS or a manual update; 3 = System driver.

wStorageType
The type of storage defined for the tag. 0 = Not stored; 1 = Cyclic; 2 = Delta; 3 = Forced storage; 17 = The storage type has been changed from cyclic to "not stored." 18 = The storage type has been changed from delta to "not stored." 19 = The storage type has been changed from "forced" to "not stored." This value is of data type smallint, with a default of 2.

wAcquisitionRate
This value is of data type int, with no default.

wStorageRate
The rate at which the tag is stored if the storage type is cyclic. The rate is in milliseconds. This value is of data type int, with a default of 1000.

wItemName
The address string of the tag. This value is of data type nvarchar(256), with a default of an empty string.

wTagType
The type of tag. 1 = Analog; 2 = Discrete; 3 = String; 4 = Complex; 5 = Event, 7 = summary tag (analog or state). This value is of data type int, with no default.

wTimeDeadband
The minimum time, in milliseconds, between stored values for a single tag. Any value changes that occur within the time deadband are not stored. The time deadband applies to delta storage only. A time deadband of 0 indicates that the system will store the value of the tag each time it changes.

CurrentEditor
Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaAnalogDetail

Returns the details for a specified analog tag, including time detector information, if applicable.

Syntax

```
aaAnalogDetail TagList
```

Arguments

TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

Permission

Execute permission defaults to the **public** group.

aaAnalogTagDelete

Deletes an analog tag.

Syntax

```
aaAnalogTagDelete wwTagKey
```

Arguments

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaAnalogTagInsert

Inserts an analog tag.

Syntax

```
aaAnalogTagInsert TagName, Description,
  AcquisitionType, StorageType, StorageRate, ItemName,
  TimeDeadband, CreatedBy, DateCreated, CurrentEditor,
  EUKey, MinEU, MaxEU, MinRaw, MaxRaw, Scaling,
  RawType, ValueDeadband, InitialValue, IntegerSize,
  SignedInteger, TopicKey, IOServerKey, StorageNodeKey,
  AIRetrievalMode, SamplesInActiveImage, RateDeadband,
  InterpolationType, RolloverValue, ServerTimeStamp,
  DeadbandType
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

Description
The description of the tag. This value is of data type nvarchar(512), with a default of an empty string.

AcquisitionType
The method by which the tag's value is acquired. If the tag value is acquired from an I/O Server, specify the name of the I/O Server, topic, and item. 0 = Not acquired; 1 = Acquired via an I/O Server; 2 = Acquired via MDAS or a manual update; 3 = System driver. This value is of data type tinyint, with a default of 1.

StorageType
The type of storage defined for the tag. 0 = Not stored; 1 = Cyclic; 2 = Delta; 3 = Forced storage; 17 = The storage type has been changed from cyclic to "not stored." 18 = The storage type has been changed from delta to "not stored." 19 = The storage type has been changed from "forced" to "not stored." This value is of data type smallint, with a default of 2.

StorageRate
The rate at which the tag is stored if the storage type is cyclic. The rate is in milliseconds. This value is of data type int, with a default of 10000.

ItemName
The address string of the tag. This value is of data type nvarchar(256), with a default of an empty string.

TimeDeadband
The minimum time, in milliseconds, between stored values for a single tag. Any value changes that occur within the time deadband are not stored. The time deadband applies to delta storage only. A time deadband of 0 indicates that the system will store the value of the tag each time it changes. This value is of data type int, with a default of 0.

CreatedBy
The name of the user or application that created the tag. This value is of data type nvarchar(256), with a default of an empty string.

DateCreated
The date that the tag was created. This value is of data type datetime, with a default of NULL.

CurrentEditor
Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server. This value is of data type int, with a default of 0.

EUKey
The unique numerical identifier of an engineering unit. This value is of data type int, with a default of 1.

MinEU
The minimum value of the tag, measured in engineering units. This value is of data type float(25), with a default of 0.

MaxEU
The maximum value of the tag, measured in engineering units. This value is of data type float(25), with a default of 100.

MinRaw
The minimum value of the raw acquired value. This value is of data type float(25), with a default of 0.

MaxRaw
The maximum value of the raw acquired value. This value is of data type float(25), with a default of 4095.

Scaling
The type of algorithm used to scale raw values to engineering units. For linear scaling, the result is calculated using linear interpolation between the end points. 0 = None; 1 = Linear; 2 = Square Root. (Square root is reserved for future use). This value is of data type int, with a default of 1.

RawType
The numeric type for the raw value. 1 = Euro Float, an outdated data type (4 bytes); 2 = MS Float (4 bytes); 3 = Integer (2 or 4 bytes); 4 = MS Double (reserved for future use) (8 bytes). This value is of data type int, with a default of 3.

ValueDeadband
The percentage of the difference between the minimum and maximum engineering units for the tag. Any data values that change less than the specified deadband are not stored. The value deadband applies to delta storage only. A value of 0 indicates that a value deadband will not be applied. This value is of data type float(25), with a default of 0.

InitialValue
The initial value as imported from an external source (for example, from InTouch). This value is of data type float(25), with a default of 0.

IntegerSize
The bit size of the analog tag. 12 = 12-bit; 15 = 15-bit; 16 = 16-bit; 32 = 32-bit; 64 = 64-bit (reserved for future use). This value is of data type tinyint, with a default of 16.

SignedInteger
Used to specify whether an integer is a signed number (positive or negative) or an unsigned number (positive only). 0 = Unsigned; 1 = Signed. This value is of data type bit, with a default of 0.

TopicKey
The unique numerical identifier for the topic. This value is of data type int, with a default of NULL.

IOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with a default of NULL.

StorageNodeKey
The unique numerical identifier for the storage node. This value is of data type int, with a default of 1.

AIRetrievalMode
Used to specify the behavior of retrieval for data in active image. You can either retrieve from all acquired data values that are currently in the active image, or only the data values that are configured to be stored on disk. Data on disk may be a subset of that in the active image, depending on the storage rate for the tag.Valid values are: 0 = All of the values received into the active image will be included in the returned data (default); 1 = Only the values that will be moved into storage will be included in the returned data. This value is of data type tinyint, with a default of 0.

SamplesInActiveImage
The number of samples that the active image holds for the tag. 0 indicates that the active image is using the default of 65 values. The higher the number of samples, the higher the load on memory resources. This value is of data type int, with a default of 0.

RateDeadband
Used to percentage of deviation in the full-scale value range for an analog tag. The swinging door (rate) deadband applies to delta storage only. Time and/or value deadbands can be used in addition to the swinging door deadband. Any value greater than 0 can be used for the deadband. A value of 0 indicates that a swinging door deadband will not be applied. This value is of data type float(25), with a default of 0.

InterpolationType
The interpolation type for retrieval. 0 = Stair-stepped interpolation; 1 = Linear interpolation (if applicable, based on the tag type); 254 = System default interpolation mode. The system default interpolation type is to use the system default for the analog type, either integer or real. The system default interpolation type for an analog type is determined by the setting of the InterpolationTypeInteger and InterpolationTypeReal system parameters. This setting impacts Interpolated, Average, and Integral retrieval modes. This value is of data type tinyint, with a default of 254.

RolloverValue
The first value that causes the counter to "roll over." This rollover value is used by the "counter" retrieval mode. For example, a counter that counts from 0 to 9999, the counter rolls over back to 0 for the 10,000th value it receives. Therefore, set the rollover value to 10,000. This value is of data type int, with a default of 0.

ServerTimeStamp
Used to specify whether local timestamping by the Wonderware Historian is used. 0 = The IDAS timestamp is used; 1 = The Wonderware Historian time is used for the timestamp. If a fast-changing tag is configured to use server timestamping, the packet of data that is sent to the storage subsystem may contain multiple data values with the same timestamp, which may affect data calculations, such as for swinging door storage. This value is of data type bit, with a default of 0.

DeadbandType
The type of delta storage deadband to be applied for the tag. This setting is only in effect if delta storage is configured for the tag. 1= Time and/or value deadband; 2 = Rate (swinging door) deadband. This value is of data type smallint, with a default of 1.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaAnalogTagSelect

Selects an analog tag.

Syntax

```
aaAnalogTagSelect wwTagKey
```

Arguments

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaAnalogTagUpdate

Updates an analog tag.

Syntax

```
aaAnalogTagUpdate wwTagKey, TagName, Description,
  AcquisitionType, StorageType, StorageRate, ItemName,
  TimeDeadband, CreatedBy, DateCreated, CurrentEditor,
  EUKey, MinEU, MaxEU, MinRaw, MaxRaw, Scaling,
  RawType, ValueDeadband, InitialValue, IntegerSize,
  SignedInteger, TopicKey, IOServerKey, StorageNodeKey,
  AIRetrievalMode, SamplesInActiveImage, RateDeadband,
  InterpolationType, RolloverValue, ServerTimeStamp,
  DeadbandType
```

Arguments

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

The remaining arguments are the same as the **aaAnalogTagInsert** stored procedure. However, only the Description, AcquisitionType, ItemName, CreatedBy, DateCreated, CurrentEditor, SamplesInActiveImage,

RateDeadband, InterpolationType, RolloverValue, ServerTimeStamp, and DeadbandType arguments have defaults.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaAnnotationDelete

Deletes an annotation.

Syntax

```
aaAnnotationDelete AnnotationKey
```

Arguments

AnnotationKey
The unique numerical identifier for the annotation. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaUsers**, **aaPowerUsers**, and **aaAdministrators** groups.

aaAnnotationInsert

Inserts an annotation.

Syntax

```
aaAnnotationInsert TagName, UserKey, DateTime,
  DateCreated, Content, Value
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

UserKey
The unique numerical identifier for a database user as defined in the UserDetail table. This value is of data type int, with a default of NULL.

DateTime
The timestamp of the tag value for which the user has made an annotation. This value is of data type datetime, with a default of NULL.

DateCreated
The date that the annotation was created. This value is of data type datetime, with a default of NULL.

Content
The annotation text. This value is of data type ntext, with a default of "Annotation."

Value
The value of the tag at the time of the annotation. This value is of data type real, with a default of 0.0.

Permission

Execute permission defaults to the **aaUsers**, **aaPowerUsers**, and **aaAdministrators** groups.

aaAnnotationRetrieve

Retrieves one or more annotations.

Syntax

```
aaAnnotationRetrieve TagList, StartTime, EndTime
```

Arguments

TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

StartTime
The starting timestamp for the data to query. This value is of data type nvarchar(50), with no default.

EndTime
The ending timestamp for the data to query. This value is of data type nvarchar(50), with no default.

Permission

Execute permission defaults to the **public** group.

aaAnnotationSelect

Selects an annotation.

Syntax

```
aaAnnotationSelect AnnotationKey
```

Arguments

AnnotationKey
The unique numerical identifier for the annotation. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaAnnotationUpdate

Updates an annotation.

Syntax

```
aaAnnotationUpdate AnnotationKey, TagName, UserKey,
  DateTime, DateCreated, Content, Value
```

Arguments

AnnotationKey
The unique numerical identifier for the annotation. This value is of data type int, with no default.

The remaining arguments are similar to those for the **aaAnnotationInsert** stored procedure.

Permission

Execute permission defaults to the **aaUsers**, **aaPowerUsers**, and **aaAdministrators** groups.

aaArchestrANSClear

Removes all ArchestrA entries from the public namespace.

Syntax

```
aaArchestrANSClear
```

Remarks

In general, using this stored procedure is not recommended. If you need to remove the ArchestrA entries because of a namespace corruption, contact Technical Support for guidance.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaCleanupAfterCommit

Runs once after reinitialization or system startup is complete.

Syntax

```
aaCleanupAfterCommit
```

Remarks

This stored procedure:

1. Sets the **DbStatus** column of the **StorageNode** table to 0.
2. Deletes the contents of the **ConfigStatusSnapshot** table.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaCleanupSystemNotRunning

Runs once whether or not reinitialization or system startup is complete. For internal use only.

Syntax

```
aaCleanupSystemNotRunning
```

Remarks

This stored procedure:

1. Sets the **DbStatus** column of the **StorageNode** table to 0.
2. Deletes information from the **ConfigStatusSnapshot** table.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaCommitChanges

Used to trigger a reinitialization of the system.

Syntax

```
aaCommitChanges
```

Remarks

This stored procedure performs the following if a change is made:

1. Copies the contents of the **ConfigStatusPending** table to the **ConfigStatusSnapshot** table.
2. Resets the **Status** column in the applicable database table (**Tag**, **Topic**, **IOServer**, **IODriver**, **StorageLocation**, **SnapshotDetail**, or **SystemParameter**) to 0.
3. Deletes the contents of the **ConfigStatusPending** table.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaCommitChangesAtStartup

Used to specify a reinitialization of the system at startup.

Syntax

```
aaCommitChangesAtStartup
```

Remarks

This stored procedure is used only when a modification to a storage location has been made. The functionality of this stored procedure is similar to that of **aaCommitChanges**.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaContextDelete

Deletes a context.

Syntax

```
aaContextDelete ContextKey
```

Arguments

ContextKey
The unique numerical identifier for the context. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers**, and **aaAdministrators** groups.

aaContextInsert

Inserts a context.

Syntax

```
aaContextInsert Description
```

Arguments

Description
The description of the context. This value is of data type nvarchar(50), with a default of NULL.

Permission

Execute permission defaults to the **aaPowerUsers**, and **aaAdministrators** groups.

aaContextSelect

Selects a context.

Syntax

```
aaContextSelect ContextKey
```

Arguments

ContextKey
The unique numerical identifier for the context. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaContextUpdate

Updates a context.

Syntax

```
aaContextUpdate ContextKey, Description
```

Arguments

ContextKey
The unique numerical identifier for the context. This value is of data type int, with no default.

The remaining argument is the same as for the **aaContextInsert** stored procedure. However, the argument does not have a default.

Permission

Execute permission defaults to the **aaPowerUsers**, and **aaAdministrators** groups.

CreateReplicationServerDefaultGroups

Used to create default replication server groups.

Syntax

```
CreateReplicationServerDefaultGroups
  ReplicationServerKey
```

Arguments

ReplicationServerKey
Unique identifier for the replication server. If specified, this will overwrite the properties of the server identified by the key.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

CreateReplicationServerSystemTags

Creates replication server default groups if the CreateGroups setting is set to true. Internal use only.

Syntax

```
CreateReplicationServerSystemTags ReplicationServerKey
```

Arguments

ReplicationServerKey
Unique identifier for the replication server. If specified, this will overwrite the properties of the server identified by the key.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaDBChangesPending

Returns a list of modifications pending, from the **ConfigStatusPending** table, in a readable format.

Syntax

```
aaDBChangesPending
```

Permission

Execute permission defaults to the **public** group.

aaDBConfig

Returns a summary of the current database configuration, such as number of tags, number of tags per type, storage configuration, event tags, summary configuration, and so on.

Syntax

```
aaDBConfig
```

Permission

Execute permission defaults to the **public** group.

aaDeleteOlderEvents

Deletes old events from event storage.

Syntax

```
aaDeleteOlderEvents
```

Remarks

This stored procedure is executed by the **aaSpaceManager** stored procedure every ten minutes. The duration for which events are kept is stored in the **SystemParameter** table. Events will be deleted from the **EventHistory** table.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaDeleteOlderSummaries

Deletes old summaries from summary storage.

Syntax

```
aaDeleteOlderSummaries
```

Remarks

This stored procedure is executed by the **aaSpaceManager** stored procedure every ten minutes. The duration for which summaries are kept is stored in the **SystemParameter** table. Summaries will be deleted from the **SummaryHistory** table.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaDeleteReplicationGroup

Delete an existing replication group. If the group being deleted is referenced by a replication tag entity, the procedure returns an error message.

Syntax

```
aaDeleteReplicationGroup GroupName,
  ReplicationServerName, ReplicationTypeKey
```

Arguments

GroupName
The name of the group. This parameter is required.

ReplicationServerName
The name of the replication server.

ReplicationTypeKey
The type of replication. Valid values are:

1 - Simple Replication

2 - Analog Summary Replication

3 - State Summary Replication

The default is 2.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaDeleteReplicationSchedule

Delete a replication schedule.

Syntax

```
aaDeleteReplicationSchedule ScheduleName
```

Arguments

ScheduleName
The name of the schedule. This parameter is required.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaDeleteReplicationServer

Delete an existing replication server. If the server being deleted is referenced by a replication group, the procedure returns an error message.

Syntax

```
aaDeleteReplicationServer ReplicationServerName
```

Arguments

ReplicationServerName
The name of the replication server. This parameter is required.

Permission

Execute permission defaults to the **aaAdministrators** group.

DeleteReplicationServerSystemTags

Deletes replication server default groups if the DeleteGroups setting is set to true. Internal use only.

Syntax

```
DeleteReplicationServerSystemTags ReplicationServerKey
```

Arguments

ReplicationServerKey
Unique identifier for the replication server. If specified, this will overwrite the properties of the server identified by the key.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaDeleteReplicationTagEntity

Delete an existing replication entity from a tier 1 server.

Syntax

```
aaDeleteReplicationTagEntity ReplicationServerName,
  DestinationTagName
```

Arguments

ReplicationServerName
The name of the replication server.

DestinationTagName
The name of the destination tag.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaDeleteTag

Deletes a tag from the database.

Syntax

```
aaDeleteTag TagName
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system.This value is of data type nvarchar(256), with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaDetectorStringSelect

Selects the detector string for a specified event tag.

Syntax

```
aaDetectorStringSelect TagName
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaDiscreteDetail

Returns the details for a specified discrete tag, including time detector information, if applicable.

Syntax

```
aaDiscreteDetail TagList
```

Arguments

TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

Permission

Execute permission defaults to the **public** group.

aaDiscreteTagDelete

Deletes a discrete tag.

Syntax

```
aaDiscreteTagDelete wwTagKey
```

Arguments

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaDiscreteTagInsert

Inserts a discrete tag.

Syntax

```
aaDiscreteTagInsert TagName, Description,
  AcquisitionType, StorageType, StorageRate, ItemName,
  TimeDeadband, CreatedBy, DateCreated, CurrentEditor,
  MessageKey, InitialValue, TopicKey, IOServerKey,
  AIRetrievalMode, SamplesInActiveImage,
  ServerTimeStamp, DeadbandType
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system.This value is of data type nvarchar(256), with no default.

Description
The description of the tag. This value is of data type nvarchar(512), with a default of an empty string.

AcquisitionType
The method by which the tag's value is acquired. If the tag value is acquired from an I/O Server, specify the name of the I/O Server, topic, and item. 0 = Not acquired; 1 = Acquired via an I/O Server; 2 = Acquired via MDAS or a manual update; 3 = System driver. This value is of data type tinyint, with a default of 1.

StorageType
The type of storage defined for the tag. 0 = Not stored; 1 = Cyclic; 2 = Delta; 3 = Forced storage; 17 = The storage type has been changed from cyclic to "not stored." 18 = The storage type has been changed from delta to "not stored." 19 = The storage type has been changed from "forced" to "not stored." This value is of data type smallint, with a default of 2.

StorageRate
The rate at which the tag is stored if the storage type is cyclic. The rate is in milliseconds. This value is of data type int, with a default of 0.

ItemName
The address string of the tag. This value is of data type nvarchar(256), with a default of an empty string.

TimeDeadband
The minimum time, in milliseconds, between stored values for a single tag. Any value changes that occur within the time deadband are not stored. The time deadband applies to delta storage only. A time deadband of 0 indicates that the system will store the value of the tag each time it changes. This value is of data type int, with a default of 0.

CreatedBy
The name of the user or application that created the tag. This value is of data type nvarchar(256), with a default of an empty string.

DateCreated
The date that the tag was created. This value is of data type datetime, with a default of NULL.

CurrentEditor
Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server. This value is of data type int, with a default of 0.

MessageKey
The unique numerical identifier of a TRUE/FALSE message pair that can be associated with a discrete tag. This value is of data type int, with a default of 1.

InitialValue
The initial value as imported from an external source (for example, from InTouch). This value is of data type tinyint, with a default of 0.

TopicKey
The unique numerical identifier for the topic. This value is of data type int, with a default of NULL.

IOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with a default of NULL.

AIRetrievalMode
Used to specify the behavior of retrieval for data in active image. You can either retrieve from all acquired data values that are currently in the active image, or only the data values that are configured to be stored on disk. Data on disk may be a subset of that in the active image, depending on the storage rate for the tag.Valid values are: 0 = All of the values received into the active image will be included in the returned data (default); 1 = Only the values that will be moved into storage will be included in the returned data. This value is of data type tinyint, with a default of 0.

SamplesInActiveImage
The number of samples that the active image holds for the tag. 0 indicates that the active image is using the default of 65 values. The higher the number of samples, the higher the load on memory resources. This value is of data type int, with a default of 0.

ServerTimeStamp
Used to specify whether local timestamping by the Wonderware Historian is used. 0 = The IDAS timestamp is used; 1 = The Wonderware Historian time is used for the timestamp. If a fast-changing tag is configured to use server timestamping, the packet of data that is sent to the storage subsystem may contain multiple data values with the same timestamp, which may affect data calculations, such as for swinging door storage. This value is of data type bit, with a default of 0.

DeadbandType
The type of delta storage deadband to be applied for the tag. This setting is only in effect if delta storage is configured for the tag. 1= Time and/or value deadband; 2 = Rate (swinging door) deadband. This value is of data type smallint, with a default of 1.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaDiscreteTagSelect

Selects a discrete tag.

Syntax

```
aaDiscreteTagSelect wwTagKey
```

Arguments

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaDiscreteTagUpdate

Updates a discrete tag.

Syntax

```
aaDiscreteTagUpdate wwTagKey, TagName, Description,
  AcquisitionType, StorageType, StorageRate, ItemName,
  TimeDeadband, CreatedBy, DateCreated, CurrentEditor,
  MessageKey, InitialValue, TopicKey, IOServerKey,
  AIRetrievalMode, SamplesInActiveImage,
  ServerTimeStamp, DeadbandType
```

Arguments

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

The remaining arguments are the same as for the **aaDiscreteTagInsert** stored procedure. However, only the Description, AcquisitionType, ItemName, CreatedBy, DateCreated, SamplesInActiveImage, ServerTimeStamp, and DeadbandType arguments have defaults.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaEngineeringUnitDelete

Deletes an engineering unit.

Syntax

```
aaEngineeringUnitDelete EUKey
```

Arguments

EUKey
The unique numerical identifier of an engineering unit. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaEngineeringUnitInsert

Inserts an engineering unit.

Syntax

```
aaEngineeringUnitInsert Unit, DefaultTagRate,
  IntegralDivisor
```

Arguments

Unit
The unit of measure. Examples are mph, grams, and pounds. This value is of data type nvarchar(32), with no default.

DefaultTagRate
The default rate, in milliseconds, at which tags are cyclically stored, based on engineering units. Although the system does not make use of this engineering unit based tag rate, you can reference this value in custom SQL scripts. The value you enter for this tag rate does not affect the default storage rate set for the tag. This value is of data type int, with a default of 10000.

IntegralDivisor

The factor to be applied when integrating a rate with the units [EngUnits/TimeUnit] to a quantity with units [EngUnits]. This factor is called the integral divisor. The default value of 1 assumes a time unit of seconds and ensures that a rate of [Unit/second] is correctly integrated to [Unit]. For a time unit of minutes, set the integral divisor value to 60; for a unit of hours, set the integral divisor value to 3600. The integral divisor is applied similarly to rates or quantities that are not expressed in terms of a time unit. For example, to convert watts to watt-hours, the integral divisor is 1/3600. To convert watts to kilowatt-hours, the integral divisor is 1/3600000. This value is of data type float(25), with a default of 1.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaEngineeringUnitSelect

Selects an engineering unit.

Syntax

```
aaEngineeringUnitSelect EUKey
```

Arguments

EUKey

The unique numerical identifier of an engineering unit. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaEngineeringUnitUpdate

Updates an engineering unit.

Syntax

```
aaEngineeringUnitUpdate EUKey, Unit, DefaultTagRate,
  IntegralDivisor
```

Arguments

EUKey
The unique numerical identifier of an engineering unit. This value is of data type int, with no default.

The remaining arguments are the same as for the **aaEngineeringUnitInsert** stored procedure. However, only the IntegralDivisor argument has a default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaEventDetection

Detects the number of events in history in which the data value for the specified tag matched the criteria defined by the remaining arguments. This stored procedure is used by the event subsystem and should not be modified.

Syntax

```
aaEventDetection TagName, Operator, DetectValue, Edge,
  Resolution, StartTime, EndTime
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

Operator
The comparison operator. Valid values are: >, >=, <, <=, =, <>. This value is of data type char(2), with no default.

DetectValue
The value against which the stored values for the tag are compared to determine if the event occurred. This value is of data type float(25), with a default of none.

Edge
The type of edge detection result set that the query will return. Valid values are NONE, LEADING, TRAILING, and BOTH. This value is of data type char(8), with no default.

Resolution
The sampling rate, in milliseconds, for retrieving the data in cyclic mode. The system returns values stored over the requested time period at the interval specified by the resolution. For example, if you specify a 5000 ms resolution, the system queries for all data during the time period and then only returns those values that occur at each 5000 ms interval, starting with the start date and ending with the end date. This value is of data type int, with no default.

StartTime
The starting timestamp for the data to query. This value is of dataype varchar(30), with no default.

EndTime
The ending timestamp for the data to query. This value is of dataype varchar(30), with no default.

Remarks

You can apply a resolution only if you set the value of the Edge argument to NONE.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaEventHistoryInsert

Inserts a row into the **EventHistory** table for each occurrence of an event for a specified event tag. This stored procedure is used by the event subsystem and should not be modified.

Syntax

```
aaEventHistoryInsert TagName, DateTime, DetectDateTime,
  Edge
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

DateTime
The timestamp reflecting when event history data was acquired. This is the time when an event actually occurred. This time reflects the time zone of the Wonderware Historian. This value is of data type datetime, with no default.

DetectDateTime
The timestamp reflecting when the event was detected by the event system. This value is of data type datetime, with no default.

Edge
The "edge" for the event detection. 0 = Trailing; 1 = Leading; 2 = Both; 3 = None; 4 = Time Detector; 5 = External Detector. This value is of data type int, with no default.

Permission
Execute permission defaults to the **aaAdministrators** group.

aaEventHistorySelect

Returns information stored in the **EventHistory** table for each specified event tag.

Syntax
```
aaEventHistorySelect TagList, StartTime, EndTime
```

Arguments
TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

StartTime
The starting timestamp for the data to query. This value is of data type nvarchar(50), with no default.

EndTime
The ending timestamp for the data to query. This value is of data type nvarchar(50), with no default.

Remarks
This stored procedure will return information for all events that occurred between the starting time and the ending time.

Permission
Execute permission defaults to the **public** group.

aaEventSnapshotInsert

Inserts snapshot values into the AnalogSnapshot, DiscreteSnapshot, and StringSnapshot tables. This stored procedure is used by the event subsystem and should not be modified.

Syntax

```
aaEventSnapshotInsert EventLogKey, EventTime,
  EventTagName
```

Arguments

EventLogKey
The unique numerical identifier of an event occurrence. This value is of data type int, with no default.

EventTime
The timestamp reflecting when the event history data was acquired. This is the time for when the event actually occurred. This value is of data type datetime, with no default.

EventTagName
The name of the event tag to which the snapshot tag is related. This value is of data type nvarchar(256), with no default.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaEventSnapshotSelect

Returns the snapshot tag values for each of the event tags specified by the tag list.

Syntax

```
aaEventSnapshot TagList, StartTime, EndTime, OrderBy
```

Arguments

TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

StartTime
The starting timestamp for the data to query. This value is of data type nvarchar(50), with no default.

EndTime
The ending timestamp for the data to query. This value is of data type nvarchar(50), with no default.

OrderBy
The name of the column in the **v_EventSnapshot** view used to order the rows in the result set. The default is 'Event'.

Remarks
This stored procedure will return information for all events that occurred between the starting time and the ending time.

This stored procedure does not work with string snapshots.

Permission
Execute permission defaults to the **public** group.

aaEventTagDelete

Deletes an event tag.

Syntax

```
aaEventTagDelete wwTagKey
```

Arguments

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaEventTagDetail

Returns the details for a specified event tag, including time detector information, if applicable.

Syntax

```
aaEventTagDetail TagList
```

Arguments

TagList
The list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with a default of '%'.

Permission
Execute permission defaults to the **public** group

aaEventTagInsert

Inserts an event tag.

Syntax

```
aaEventTagInsert TagName, Description, CreatedBy,
  DateCreated, CurrentEditor, ScanRate, TimeDeadband,
  Logged, Status, PostDetectorDelay, UseThreadPool,
  DetectorTypeKey, DetectorString, ActionTypeKey,
  ActionString, Priority, Edge
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

Description
The description of the tag. This value is of data type nvarchar(512), with a default of an empty string.

CreatedBy
The name of the user or application that created the tag. This value is of data type nvarchar(256), with a default of an empty string.

DateCreated
The date that the tag was created. This value is of data type datetime, with a default of NULL.

CurrentEditor
Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server. This value is of data type int, with a default of 0.

ScanRate
The interval, in milliseconds, at which the system checks to see if the event conditions specified by the detector occurred. This value must be greater than or equal to 500 milliseconds, and less than or equal to 1 hour (3600000 ms). This value is of data type int, with a default of 0.

TimeDeadband
The minimum time, in milliseconds, between stored values for a single tag. Any value changes that occur within the time deadband are not stored. The time deadband applies to delta storage only. A time deadband of 0 indicates that the system will store the value of the tag each time it changes. This value is of data type int, with a default of 0.

Logged
Used to specify whether or not to log events for this tag into the EventHistory table. Event logging can only be turned off if no associated actions are configured. 0 = Not logged; 1 = Logged. This value is of data type bit, with a default of 0.

Status
The flag used by the event system at system startup and during runtime to determine if the event tag has been modified. 0 = Posted. Any changes have been detected and effected by the system. 1 = New. An event tag has been inserted, but is not yet executing. 2 = Modification. An event tag has been updated, but the older one is already executing. 98 = Disabled. 99 = Disabling requested. The event tag does not execute, even though the definition still exists in the schema. Note that there may be a delay of up to 30 seconds before a change in an event tag is seen by the running system. This value is of data type tinyint, with a default of 0.

PostDetectorDelay
The amount of time, in milliseconds, that must elapse after an event is detected before the event action can be executed. This value is of data type int, with a default of 0.

UseThreadPool
To specify how system threads are used to process events. 1 = All events are handled by a single thread and a single logon to the SQL Server; 0 = Each event uses a separate system thread and logon. This will allow the event subsystem to manage the scan rates of each detector component concurrently. (Reserved for future use.) This value is of data type bit, with a default of 1.

DetectorTypeKey
The unique identifier of a particular type of detector. Event tags and detectors are linked by means of this key. The event system relies on the following values, which are added during installation: 1 = System; 2 = External event; 3 = Generic SQL; 4 = Analog specific value; 5 = Discrete specific value; 6 = Time-based (schedule). This value is of data type int, with a default of 0.

DetectorString
The script that contains the criteria for event detection. Detector scripts are executed on the local Wonderware Historian. This value is of data type nvarchar(1500), with a default of NULL.

ActionTypeKey
The unique identifier for a particular type of action. Event tags and actions are linked by this key. The event subsystem relies on the following values, which are added during installation: 1 = No action; 2 = Generic SQL; 3 = Snapshot; 4 = E-mail; 5 = Deadband; 6 = Summary. This value is of data type int, with a default of 0.

ActionString
The script that specifies the event action. Action scripts run on the local Wonderware Historian. This value is of data type nvarchar(1500), with a default of NULL.

Priority
The priority level for the action, either critical or normal. The priority level determines the sorting queue to which the action will be sent. The critical queue is used for highly important events. If a system overload condition occurs, events that are given a critical priority will always be processed first. Events that are given a normal priority will be processed after any critical events and may possibly be dropped (that is, not performed) on an overloaded system. This value is of data type tinyint, with a default of 0.

Edge
The "edge" for the event detection. 0 = Trailing; 1 = Leading; 2 = Both; 3 = None; 4 = Time Detector; 5 = External Detector. This value is of data type tinyint, with a default of 1.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaEventTagSelect

Selects an event tag.

Syntax

```
aaEventTagSelect wwTagKey
```

Arguments

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaEventTagSelectAll

Used by the event system to determine changes for dynamic reinitialization.

Syntax

```
aaEventTagSelectAll
```

Permission

Execute permission defaults to the **public** group.

aaEventTagSelectDeleted

Used by the event system to determine changes for dynamic reinitialization.

Syntax

```
aaEventTagSelectDeleted
```

Permission

Execute permission defaults to the **public** group.

aaEventTagSelectDisabled

Used by the event system to determine changes for dynamic reinitialization.

Syntax

```
aaEventTagSelectDisabled
```

Permission

Execute permission defaults to the **public** group.

aaEventTagSelectInserted

Used by the event system to determine changes for dynamic reinitialization.

Syntax

```
aaEventTagSelectInserted
```

Permission

Execute permission defaults to the **public** group.

aaEventTagSelectUpdated

Used by the event system to determine changes for dynamic reinitialization.

Syntax

```
aaEventTagSelectUpdated
```

Permission

Execute permission defaults to the **public** group.

aaEventTagUpdate

Updates an event tag.

Syntax

```
aaEventTagUpdate wwTagKey, TagName, Description,
  CreatedBy, DateCreated, CurrentEditor, ScanRate,
  TimeDeadband, Logged, Status, PostDetectorDelay,
  UseThreadPool, DetectorTypeKey, DetectorString,
  ActionTypeKey, ActionString, Priority, Edge
```

Arguments

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

The remaining arguments are the same as for the **aaEventTagInsert** stored procedure. However, wwTagKey, TagName, Description, DateCreated, DetectorString, and ActionString do not have defaults.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaGetAnalogSummaryTags

Returns all the properties for the specified analog summary tag or if you don't specify a tag, returns this info for them all.

Syntax

```
aaGetAnalogSummaryTags TagName
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

Permission

Execute permission defaults to the **public** group.

aaGetDbRevision

Used to determine the current revision number of the database.

Syntax

```
aaGetDbRevision
```

Permission

Execute permission defaults to the **public** group.

aaGetLastTagKey

Returns the details for the last inserted tag.

Syntax

```
aaGetLastTagKey TagType
```

Arguments

TagType
The type of tag. 1 = Analog; 2 = Discrete; 3 = String; 4 = Complex; 5 = Event, 7 = summary tag (analog or state). This value is of data type int, with no default.

Permission

Execute permission defaults to the **public** group.

aaGetReplicationGroups

Returns the groups configured in the Historian database for a given replication server and type. If you specify all the parameters, then the specific group identified by the parameters is returned.

Syntax

```
aaGetReplicationGroups ConfigurationToReturn,
  ReplicationServerName, ReplicationTypeKey,
  ReplicationGroupName, IncludeEmptyGroups,
  ReplicationGroupKey, RowsToReturn
```

Arguments

ConfigurationToReturn
The return configuration for the replication service.This value is of data type int, with a default of 1.

ReplicationServerName
The name of the replication server. This value is nvarchar(255), with a default of NULL.

ReplicationTypeKey
The type of replication. Value values are:

1 - Simple Replication

2 - Analog Summary Replication

3 - State Summary Replication

This value is of data type int, with a default of 2.

ReplicationGroupName
The name of the replication group. This value is of data type int, with a default of NULL.

IncludeEmptyGroups
Bit that specifies whether to include empty groups in the return. This value is of data type bit, with a default of 0.

ReplicationGroupKey
Unique identifier for the replication group. This value is of data type int, with a default of NULL.

RowsToReturn
The number of rows to return. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **aaUsers**, **aaPowerUsers**, and **aaAdministrators** groups.

aaGetReplicationNamingParameters

Returns the naming parameters for the specified replication type in the specified replication group.

Syntax

```
aaGetReplicationNamingParameters ReplicationTypeKey,
  ReplicationGroupKey
```

Arguments

ReplicationTypeKey
The type of replication. Value values are:

1 - Simple Replication

2 - Analog Summary Replication

3 - State Summary Replication

This value is of data type tinyint, with a default of 3.

ReplicationGroupKey
Unique identifier for the replication group. This value is of data type int, with a default of NULL..

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaGetReplicationSchedules

Returns the schedules configured in the Historian database.

Syntax

```
aaGetReplicationSchedules ConfigurationToReturn,
  ReplicationScheduleName, ReplicationScheduleKey,
  RowsToReturn
```

Arguments

ConfigurationToReturn
The return configuration for the replication schedule. This value is of data type int, with a default of 0.

ReplicationScheduleName
The name of the schedule. This value is nvarchar(255), with a default of NULL.

ReplicationScheduleKey
The unique identifier for the schedule. This value is of data type int, with a default of NULL.

RowsToReturn
The number of rows to return. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **aaUsers**, **aaPowerUsers**, and **aaAdministrators** groups.

aaGetReplicationServers

Returns the configured replication servers in the database. If the server name is specified, then it will return only the properties of the server identified by the name.

Syntax

```
aaGetReplicationServers ConfigurationToReturn,
  ReplicationServerName, ReplicationServerKey,
  RowsToReturn
```

Arguments

ConfigurationToReturn
Returns teh configuration for the replication service. This value is of data type int, with a default of 0.

ReplicationServerName
The name of the server. This value is nvarchar(255), with a default of NULL.

ReplicationServerKey
The unique identifier for the server. This value is of data type int, with a default of NULL.

RowsToReturn
The number of rows to return. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **aaUsers**, **aaPowerUsers**, and **aaAdministrators** groups.

aaGetReplicationTagEntities

Returns the replication entities configured in the Historian database. This procedure will return the properties of the replication tag entity based on the following parameter order:

- If a ReplicationTagEntityKey is specified, then the specific entity properties are returned.
- If a SourceTagName is specified, then all the entities with the specific SourceTagName are returned.
- If the Replication Server and GroupName are not specified, then all the entities belonging to the specific replication type are returned.
- If the Replication Server and GroupName and type are specified, then all the entities belonging to the specific group and type are returned.

Syntax

```
aaGetReplicationTagEntities ConfigurationToReturn,
  ReplicationServerName, ReplicationGroupName,
  ReplicationTypeKey, SourceTagName,
  ReplicationTagEntityKey, RowsToReturn, FetchModified
```

Arguments

ConfigurationToReturn
The return configuration for the replication entites.This value is of data type int, with a default of 1.

ReplicationServerName
The name of the server. This value is nvarchar(255), with a default of NULL.

ReplicationGroupName
The name of the replication group. This value is nvarchar(255), with a default of NULL.

ReplicationScheduleKey
The unique identifier for the schedule. This value is of data type int, with a default of NULL.

ReplicationTypeKey
The type of replication. Value values are:

1 - Simple Replication

2 - Analog Summary Replication

3 - State Summary Replication

This value is of data type tinyint, with a default of 2.

SourceTagName
The name of the source tag. This value is of data type nvarchar(256), with a default of NULL.

ReplicationTagEntityKey
The unique identifier for the replication tag entity. This value is of data type int, with a default of NULL.

RowsToReturn
The number of rows to return. This value is of data type int, with a default of 3.

FetchModified
Returns requested entities. Valid values are:

1 = fetch only modified entities

0 = fetch all

Permission

Execute permission defaults to the **aaUsers**, **aaPowerUsers**, and **aaAdministrators** groups.

aaGetReplicationTags

Returns the specified replication tag.

Syntax

```
aaGetReplicationTags TagName
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with a default of NULL.

Permission

Execute permission defaults to the public group.

aaGetStateSummaryTags

Returns the specified state summary tag.

Syntax

```
aaGetStateSummaryTags TagName
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

Permission

Execute permission defaults to the public group.

aaHistorianConfigNSExpand

Expands the tree view under an Wonderware Historian in the namespace. This stored procedure is used by the Configuration Editor component of the System Management Console and should not be modified.

Syntax

```
aaHistorianConfigNSExpand PKey
```

Arguments

PKey
A local variable used to identify the Wonderware Historian in the namespace. This value is of data type int, with no default.

Remarks

An object can have one or more objects below it in the namespace hierarchy.

Permission

Execute permission defaults to the **public** group.

aaHistorianNSExpand

Expands the tree view under an Wonderware Historian in the namespace.

Syntax

```
aaHistorianNSExpand PKey
```

Arguments

PKey
A local variable used to identify the Wonderware Historian in the namespace. This value is of data type int, with no default.

Permission

Execute permission defaults to the **public** group.

aaHistorianStatusSelect

Used to select the value of the database status flag, DbStatus.

Syntax

```
aaHistorianStatusSelect
```

Remarks

This stored procedure is used by the System Management Console to determine the state of a database modification.

Permission

Execute permission defaults to the **public** group.

aaHistorianStatusSet

Sets the value of the status flag, DbStatus, to a value greater than 0 when a database modification needs to be processed by the server (back end). Sets the value of DbStatus back to 0 when a database modification is complete.

Syntax

```
aaHistorianStatusSet DbStatus, Acquisition, Storage,
  DBServer
```

Arguments

DbStatus

For releases prior to 8.0, used to store the status of server reinitializations. 2 = Certain columns in the Tag, AnalogTag, DiscreteTag, StringTag, Topic, and IOServer tables were changed; 3 = Reinitialization needed; 4 = Commit phase of a database update is in progress; 0 = Reinitialization complete. A negative value indicates that an error was encountered during reinitialization. This value is of data type int, with no default.

Acquisition

Used with DbStatus to indicate to the back end whether the acquisition subsystem needs to be restarted. 0 = Restart not needed; 1 = Restart needed. Currently not used. This value is of data type int, with a default of 0.

Storage

Used with DbStatus to indicate to the back end whether the storage subsystem needs to be restarted. 0 = Restart not needed; 1 = Restart needed. Currently not used. This value is of data type int, with a default of 0.

DBServer

Used with DbStatus to indicate to the back end whether the database server needs to be restarted. 0 = Restart not needed; 1 = Restart needed. Currently not used. This value is of data type int, with a default of 0.

Note Only the first argument is required; the others are used to indicate that a specific sub-system needs to be initialized.

Remarks

When a change is made to the Runtime database configuration using the System Management Console, the value of the DbStatus attribute in the StorageNode table is set to a value greater than 0, meaning that modifications are outstanding and a reininitialization has yet to occur. The System Management Console, after detecting that a change is outstanding based on the value of DbStatus, makes the required changes, reinitializes the Wonderware Historian, if necessary, and then sets the value of DbStatus back to 0, meaning that reinitialization has been completed.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaHistoryBlockSelect

Returns the list of history blocks for the selected time period. If no arguments are passed, the complete list is returned.

Syntax

```
aaHistoryBlockSelect FromDate, ToDate
```

Arguments

FromDate

The starting timestamp for the history block. This value is of data type datetime, with a default of NULL.

ToDate

The ending timestamp for the history block. This value is of data type datetime, with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaInTouchNodeTagList

Used by the System Management Console to display a list of imported tags for an InTouch node.

Syntax

```
aaInTouchNodeTagList NodeKey, FilterStr
```

Arguments

NodeKey

The unique numerical identifier of the named InTouch node. This value is of data type int, with a default of NULL.

FilterStr

Used to order the tagnames in the namespace. This value is of data type nvarchar(100), with a default of NULL.

Remarks

This stored procedure returns the Wonderware Historian tagname, the original InTouch tagname, and the InTouch tag type (for example, memory integer).

Permission

Execute permission defaults to the **public** group.

aaIODriverDelete

Deletes an IDAS.

Syntax

```
aaIODriverDelete IODriverKey
```

Arguments

IODriverKey

The unique identifier for an IDAS. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaAdministrators** and **aaPowerUsers** groups.

aaIODriverInsert

Inserts an IDAS.

Syntax

```
aaIODriverInsert StorageNodeKey, ComputerName,
  StoreForwardMode, StoreForwardPath, MinMBThreshold,
  AltComputerName, Enabled, StoreForwardDuration,
  AutonomousStartupTimeout, BufferCount, FileChunkSize,
  ForwardingDelay, ConnectionTimeout
```

Arguments

StorageNodeKey

The unique numerical identifier for the storage node. This value is of data type int, with a default of 1.

ComputerName

The name of the computer on which the IDAS runs. This value is of data type nvarchar(255), with a default of the name of the local server running Microsoft® SQL Server.

StoreForwardMode

Used to specify whether or not store-and-forward capability is enabled. If enabled, and the network connection between the IDAS and the storage node fails, data will be "buffered" to the location specified by the store-and-forward path. Valid values are: 0 = Disabled; 1 = Enabled; 2 = Autonomous. The Autonomous mode (2) is an extension of the normal store-and-forward mode (1). It allows the IDAS to start up using an IDAS configuration file and collect data in store-and-forward mode if the network connection to the Wonderware Historian is not available. This value is of data type tinyint, with a default of 0.

StoreForwardPath

Used to specify the path for the IDAS data buffer on the local hard drive of the IDAS computer. The path should be absolute (for example, c:\IDASBuffer). Data is written to this path until the minimum threshold for the buffer is reached. Remote buffer paths are not supported. When the store-and-forward path specified for the IDAS is invalid, the default path picked by the system is: <public folder>\ArchestrA\Historian\IDAS\SF where the <public folder> is dependent on the operating system. For example, for the Windows 2008 and Windows Vista operating systems, the path is C:\ProgramData\ArchestrA\Historian\IDAS\SF. On the Windows 2003 and Windows XP operating systems, the path is C:\Documents and Settings\All Users\Application Data\ArchestrA\Historian\IDAS\SF. When the store-and-forward path specified for the IDAS is just a folder name (without any path characters like \ and :), the default path picked by the system is: <Windows system path>\<folder name specified by the user>. For example, for the Windows Server 2003, Windows XP, Windows Vista 32-bit, and Windows Server 2008 32-bit operating systems, the path is C:\WINDOWS\system32\<folder name>. This value is of data type nvarchar(255), with a default of an empty string.

MinMBThreshold

The minimum amount of free disk space, in megabytes, at which IDAS stops collecting data in the store-and-forward buffer. This value is of data type int, with a default of 16.

AltComputerName
The name of the computer on which an optional, redundant IDAS runs. You must use the fully qualified name of the computer. You could also use the IP address. This should be set to an empty string if no redundant IDAS is specified. Make sure that the IDAS software is installed on the target failover computer. If the failure of the primary IDAS is detected by the system, the failover IDAS is automatically started. The failover IDAS is shut down after the primary IDAS is back online. This value is of data type nvarchar(255), with a default of an empty string.

Enabled
Used to specify whether the IDAS is enabled or not. 0 = Not enabled; 1 = enabled. Disabling the IDAS allows for the configuration to be retained in the database, even though the IDAS is removed from the system. This value is of data type bit, with a default of 1.

StoreForwardDuration
The minimum duration, in seconds, for the IDAS to function in store-and-forward mode. The IDAS functions in store-and-forward mode for this length of time even if the condition that caused IDAS to function in store-and-forward mode no longer exists. The maximum duration is 3600 seconds, and the minimum is 0 seconds. This value is of data type int, with a default of 180.

AutonomousStartupTimeout
The amount of time, in seconds, that the autonomous IDAS should wait for configuration commands when started by the Configuration service before going to the autonomous mode. This timeout may need to be increased only if you have a large number of IDASs configured as autonomous on a slow network. This value is of data type int, with a default of 60.

BufferCount
The number of 64 KB buffers pre-allocated for buffering data. This number may need to be increased to accommodate high data rates. This value is of data type int, with a default of 128.

FileChunkSize
The size, in bytes, of the data "chunks" that are sent to the historian when store-and-forward data is forwarded. The size of the chunks can be decreased to accommodate slower networks. Decrease this number only if the forwarding delay is greater than zero. This value is of data type int, with a default of 65536.

ForwardingDelay
The interval, in milliseconds, at which "chunks" of store-and-forward data are forwarded to the historian. The length of the interval may need to be increased to accommodate slower networks. This value is of data type int, with a default of 0.

ConnectionTimeout
The amount of time, in seconds, that the Configuration service attempts to communicate with an IDAS for configuration/reconfiguration. If this timeout elapses, the Configuration service assumes that the IDAS connection has been dropped. This number may need to be increased to accommodate slower networks. This value is of data type int, with a default of 30.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaIODriverSelect

Selects an IDAS.

Syntax

```
aaIODriverSelect IODriverKey
```

Arguments

IODriverKey
The unique identifier for an IDAS. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaIODriverUpdate

Updates an IDAS.

Syntax

```
aaIODriverUpdate IODriverKey, StorageNodeKey,
  ComputerName, StoreForwardMode, StoreForwardPath,
  MinMBThreshold, AltComputerName, Enabled,
  StoreForwardDuration, AutonomousStartupTimeout,
  BufferCount, FileChunkSize, ForwardingDelay,
  ConnectionTimeout
```

Arguments

IODriverKey
The unique identifier for an IDAS. This value is of data type int, with no default.

The remaining arguments are the same as for the **aaIODriverInsert** stored procedure. However, only StorageNodeKey, MinMBThreshold, Enabled, StoreForwardDuration, AutonomousStartupTimeout, BufferCount, FileChunkSize, ForwardingDelay, and ConnectionTimeout have defaults.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaIOServerDelete

Deletes an I/O Server from the system configuration.

Syntax

```
aaIOServerDelete IOServerKey
```

Arguments

IOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaIOServerInsert

Inserts an I/O Server into the system configuration.

Syntax

```
aaIOServerInsert StorageNodeKey, ApplicationName,
  Description, Path, ComputerName, AutoStart, ExeType,
  InitializationStatus, ProtocolType, AltComputerName,
  IODriverKey
```

Arguments

StorageNodeKey
The unique numerical identifier for the storage node. This value is of data type int, with a default of 1.

ApplicationName
The application name of the I/O Server. This name is usually the same as the executable file name. This value is of data type nvarchar(32), with no default.

Description
The description of the I/O Server. This value is of data type nvarchar(50), with a default of NULL.

Path
The full UNC path (including the filename) to locate the executable file for the I/O Server. If the I/O Server type key is specified, the filename may be omitted. This value is nvarchar(255), with a default of NULL.

ComputerName
The name of the computer on which the I/O Server runs.This value is of data type nvarchar(255), with no default.

AutoStart
Used to control how the I/O Server starts up. 0 = Automatic startup when the system starts. 1 = Manual startup required. Currently not used. This value is of data type bit, with a default of 0.

ExeType
The type of executable for the I/O Server. Used by the Historian System Management Console to determine how to start the I/O Server. 0 = Service; 1 = Console application; 2 = Windows application. This value is of data type int, with a default of 0.

InitializationStatus
The control flag used to ensure that each I/O Server has been asked for the data type (integer or real) of each tag that it will send. Only needed after a database modification. This value is of data type tinyint, with no default.

ProtocolType
The protocol used by the Wonderware Historian server to communicate with the I/O Server. 1 = DDE; 2 = SuiteLink™; 3 = Wonderware Historian named pipe driver (for compatibility with IndustrialSQL Server 3.0 and previous versions). Of the operating systems currently supported by the Wonderware Historian, DDE is only supported on the Windows XP operating system. This value is of data type int, with a default of 1.

AltComputerName
The name of the computer on which an optional, failover I/O Server runs. The failover I/O Server must be running in order for the switch to be made. This value is nvarchar(255), with a default of NULL.

IODriverKey
The unique identifier for an IDAS. This value is of data type int, with a default of 2.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaIOServerSelect

Selects an I/O Server from the system configuration.

Syntax

```
aaIOServerSelect IOServerKey
```

Arguments

IOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaIOServerTypeDelete

Deletes an I/O Server type from the system configuration.

Syntax

```
aaIOServerTypeDelete ApplicationName
```

Arguments

ApplicationName
The application name of the I/O Server. This name is usually the same as the executable file name. This value is of data type nvarchar(32), with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaIOServerTypeInsert

Inserts an I/O Server type into the system configuration.

Syntax

```
aaIOServerTypeInsert ApplicationName, Description,
  ExeName, Revision, Platform
```

Arguments

ApplicationName
The application name of the I/O Server. This name is usually the same as the executable file name. This value is of data type nvarchar(32), with no default.

Description
The description of the I/O Server type. This value is of data type nvarchar(50), with a default of NULL.

ExeName
The name of the I/O Server's executable file. This value is nvarchar(255), with a default of NULL.

Revision
The revision number for the I/O Server. This value is of data type nchar(20), with a default of NULL.

Platform
The operating system required by the I/O Server. Valid operating systems are: WINDOWS NT, WINDOWS 95, WINDOWS 98, WINDOWS XP, WINDOWS 2000, WINDOWS 2003, WINDOWS XP, WINDOWS VISTA. This value is of data type nchar(20), with a default of NULL.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaIOServerTypeSelect
Selects an I/O Server type from the system configuration.

Syntax
```
aaIOServerTypeSelect ApplicationName
```

Arguments
ApplicationName
The application name of the I/O Server. This name is usually the same as the executable file name. This value is of data type nvarchar(32), with a default of NULL.

Permission
Execute permission defaults to the **public** group.

aaIOServerTypeUpdate
Updates an I/O Server type in the system configuration.

Syntax
```
aaIOServerTypeUpdate ApplicationName, Description,
  ExeName, Revision, Platform
```

Arguments
All arguments are the same as for the **aaIOServerTypeInsert** stored procedure. However, none of the arguments have defaults.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaIOServerUpdate

Updates an I/O Server in the system configuration.

Syntax

```
aaIOServerUpdate IOServerKey, StorageNodeKey,
  IODriverKey, ApplicationName, Description, Path,
  ComputerName, AutoStart, ExeType,
  InitializationStatus, ProtocolType, AltComputerName
```

Arguments

IOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with no default.

The remaining arguments are the same as for the **aaIOServerInsert** stored procedure. However, only the AltComputerName argument has a default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaLimitDelete

Deletes a limit.

Syntax

```
aaLimitDelete TagName, ContextKey, LimitNameKey
```

Arguments

All arguments are the same as for the **aaLimitInsert** stored procedure. However, none of the arguments have defaults.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaLimitInsert

Inserts a limit.

Syntax

```
aaLimitInsert TagName, ContextKey, LimitType, Value,
  LimitNameKey, Priority, Checked, Description
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system.This value is of data type nvarchar(256), with no default.

ContextKey
The unique numerical identifier for the context. This value is of data type int, with a default of 1.

LimitType
The type of limit; that is, whether it is a rising (up) or falling (down) limit. 0 = Rising; 1 = Falling. This value is of data type int, with a default of 1.

Value
The value that is used as a specific limit for a tag. In theory, a tag can have an infinite number of limits defined. This value is of data type real, with no default.

LimitNameKey
The unique numerical identifier associated with a limit name. This value is of data type int, with no default.

Priority
The priority for the limit. Priorities can range from 1 to over 2 billion, with 1 being the highest priority. This value is of data type int, with a default of 1.

Checked
Used to specify whether a tag imported from InTouch is configured for automatic limit checking. Only checked limits are imported. 0 = Checking disabled; 1 = Checking enabled. This value is of data type bit, with a default of 1.

Description
The description of the limit. This value is of data type nvarchar(50), with a default of NULL.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaLimitNameDelete

Deletes a limit name.

Syntax

```
aaLimitNameDelete LimitNameKey
```

Arguments

LimitNameKey
The unique numerical identifier associated with a limit name. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaLimitNameInsert

Inserts a limit name.

Syntax

```
aaLimitNameInsert Name
```

Arguments

Name
The name for the limit. This value is of data type nvarchar(20), with a default of an empty string.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaLimitNameSelect

Selects a limit name.

Syntax

```
aaLimitNameSelect LimitNameKey
```

Arguments

LimitNameKey
The unique numerical identifier associated with a limit name. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaLimitNameUpdate

Updates a limit name.

Syntax

```
aaLimitNameUpdate LimitNameKey, Name
```

Arguments

LimitNameKey
The unique numerical identifier associated with a limit name. This value is of data type int, with no default.

Name
The name for the limit. This value is of data type nvarchar(20), with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaLimitSelect

Selects a limit.

Syntax

```
aaLimitSelect TagName
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. The limit will be selected for the specified tag. This value is of data type nvarchar(256), with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaLimitUpdate

Updates a limit.

Syntax

```
aaLimitUpdate TagName, ContextKey, LimitType, Value,
  LimitNameKey, Priority, Checked, Description
```

Arguments

All arguments are the same as for the **aaLimitInsert** stored procedure. However, only the Description argument has a default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaMessageDelete

Deletes a message for a discrete tag.

Syntax

```
aaMessageDelete MessageKey
```

Arguments

MessageKey
The unique numerical identifier of a TRUE/FALSE message pair that can be associated with a discrete tag. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaMessageInsert

Inserts a message for a discrete tag.

Syntax

```
aaMessageInsert Message0, Message1
```

Arguments

Message0
The message associated with the FALSE state of the discrete tag. The maximum number of characters is 64. A discrete tag set to 0 is in the FALSE state. This value is of data type nvarchar(64), with a default of NULL.

Message1
The message associated with the TRUE state of the discrete tag. The maximum number of characters is 64. A discrete tag set to 1 is in the TRUE state. This value is of data type nvarchar(64), with a default of NULL.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaMessageSelect

Selects a message for a discrete tag.

Syntax

```
aaMessageSelect MessageKey
```

Arguments

MessageKey
The unique numerical identifier of a TRUE/FALSE message pair that can be associated with a discrete tag. This value is of data type int, with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaMessageUpdate

Updates a message for a discrete tag.

Syntax

```
aaMessageUpdate MessageKey, Message0, Message1
```

Arguments

MessageKey
The unique numerical identifier of a TRUE/FALSE message pair that can be associated with a discrete tag. This value is of data type int, with no default.

The remaining arguments are the same as for the aaMessageInsert stored procedure. However, none of the arguments have defaults.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaModLogStatus

Used to retrieve the status of modification tracking.

Syntax

```
aaModLogStatus
```

Remarks

This stored procedure is used by the System Management Console. Modification tracking is controlled by the value of the ModLogTrackingStatus system parameter, which is stored in the Value column of the SystemParameter table. If the value of this column is set to a value from 1 to 7, then modification tracking is on (0 = off).

Permission

Execute permission defaults to the **public** group.

aaPrivateNSAddGroup

Adds a group object in the private namespace under the specified parent object in the namespace hierarchy.

Syntax

```
aaPrivateNSAddGroup Name, ParentKey, Type
```

Arguments

Name

The name of this object in the hierarchy. This value is of data type nvarchar(255), with no default.

ParentKey

The unique identifier for a named object in this namespace. This value is of data type int, with no default.

Type

The value that specifies the type of namespace. 1 to 6 = Tag; 1 to 2 million = System; 2+ million = Groups. This value is of data type int, with a default of 1000000.

Permission

Execute permission defaults to the **public** group.

aaPrivateNSAddLeaf

Adds a single object in the private namespace under the currently selected object in the namespace hierarchy.

Syntax

```
aaPrivateNSAddLeaf wwTagKey, NameKey, ServerKey
```

Arguments

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

NameKey
The unique identifier for the object in the namespace. This value is of data type int, with no default.

ServerKey
The unique numerical identifier of a Wonderware Historian server. This value is of data type int, with a default of 1.

Permission

Execute permission defaults to the **public** group.

aaPrivateNSDeleteGroup

Deletes a group object, as well as any objects under it, in the private namespace.

Syntax

```
aaPrivateNSDeleteGroup NameKey
```

Arguments

NameKey
The unique identifier for the object in the namespace. This value is of data type int, with no default.

Permission

Execute permission defaults to the **public** group.

aaPrivateNSDeleteLeaf

Deletes a single object in the private namespace.

Syntax

```
aaPrivateNSDeleteLeaf NameKey, wwTagKey
```

Arguments

NameKey
The unique identifier for the object in the namespace. This value is of data type int, with no default.

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

Permission

Execute permission defaults to the **public** group.

aaPrivateNSExpand

Expands the tree view one level under a single parent object in the private namespace.

Syntax

```
aaPrivateNSExpand PKey, FilterStr
```

Arguments

PKey
A local variable used to identify the object in the namespace. This value is of data type int, with no default.

FilterStr
Used to order the tagnames in the namespace. This value is of data type nvarchar(100), with a default of NULL.

Remarks

A parent object can have one or more objects below it in the namespace hierarchy.

Permission

Execute permission defaults to the **public** group.

aaPrivateNSSelect

Selects all valid group objects (items) for the current user in the private namespace.

Syntax

```
aaPrivateNSSelect
```

Permission

Execute permission defaults to the **public** group.

aaPrivateNSUpdateGroup

Updates a group object in the private namespace.

Syntax

```
aaPrivateNSUpdateGroup NameKey, Name, Type
```

Arguments

NameKey
The unique identifier for the object in the namespace. This value is of data type int, with no default.

Name
The name of this object in the hierarchy. This value is of data type nvarchar(255), with no default.

Type
The value that specifies the type of namespace. 1 to 6 = Tag; 1 to 2 million = System; 2+ million = Groups. This value is of data type int, with no default.

Permission

Execute permission defaults to the **public** group.

aaPublicNSAddGroup

Adds a group object in the public namespace under the specified parent object in the namespace hierarchy.

Syntax

```
aaPublicNSAddGroup Name, ParentKey, Type
```

Arguments

Name
The name of this object in the hierarchy. This value is of data type nvarchar(255), with no default.

ParentKey
The unique identifier for a named object in this namespace. This value is of data type int, with no default.

Type
The value that specifies the type of namespace. 1 to 6 = Tag; 1 to 2 million = System; 2+ million = Groups. Within the system range, the following values designate ArchestrA object types: 1999023 = Galaxy; 1999001 = WinPlatform object; 1999003 = AppEngine object; 1999013 = Area object; 1999011 = DDESuiteLinkClient, OPCClient, and InTouchProxy objects; 1999024 = RedundantDIObject object; 1999033 = Undeployed object represented by a generic name; 1999901 = ApplicationObject; 1999902 = Traceability object. This value is of data type int, with a default of 1000000.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaPublicNSAddLeaf
Adds a single object in the public namespace under the currently selected object in the namespace hierarchy.

Syntax
```
aaPublicNSAddLeaf wwTagKey, NameKey, ServerKey
```

Arguments
wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

NameKey
The unique identifier for the object in the namespace. This value is of data type int, with no default.

ServerKey
The unique numerical identifier of a Wonderware Historian server. This value is of data type int, with a default of 1.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaPublicNSDeleteGroup

Deletes a group object, as well as any objects under it, in the public namespace.

Syntax

```
aaPublicNSDeleteGroup NameKey
```

Arguments

NameKey
The unique identifier for the object in the namespace. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaPublicNSDeleteLeaf

Deletes a single object in the public namespace.

Syntax

```
aaPublicNSDeleteLeaf NameKey, wwTagKey
```

Arguments

NameKey
The unique identifier for the object in the namespace. This value is of data type int, with no default.

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaPublicNSExpand

Expands the tree view one level under a single parent object in the public namespace.

Syntax

```
aaPublicNSExpand PKey, FilterStr
```

Arguments

PKey
A local variable used to identify the object in the namespace. This value is of data type int, with no default.

FilterStr
Used to order the tagnames in the namespace. This value is of data type nvarchar(100), with a default of NULL.

Remarks

A parent object can have one or more objects below it in the namespace hierarchy.

Permission

Execute permission defaults to the **public** group.

aaPublicNSSelect

Selects all valid group objects (items) in the public namespace.

Syntax

```
aaPublicNSSelect
```

Permission

Execute permission defaults to the **public** group.

aaPublicNSUpdateGroup

Updates a group object in the public namespace.

Syntax

```
aaPublicNSUpdateGroup NameKey, Name, Type
```

Arguments

NameKey
The unique identifier for the object in the namespace. This value is of data type int, with no default.

Name
The name of this object in the hierarchy. This value is of data type nvarchar(255), with no default.

Type
The value that specifies the type of namespace. 1 to 6 = Tag; 1 to 2 million = System; 2+ million = Groups. Within the system range, the following values designate ArchestrA object types: 1999023 = Galaxy; 1999001 = WinPlatform object; 1999003 = AppEngine object; 1999013 = Area object; 1999011 = DDESuiteLinkClient, OPCClient, and InTouchProxy objects; 1999024 = RedundantDIObject object; 1999033 = Undeployed object represented by a generic name; 1999901 = ApplicationObject; 1999902 = Traceability object. This value is of data type int, with no default.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaRedirectToInTouch
Redirects the tag address (item name) to the InTouch node, rather than to the original I/O Server.

Syntax
```
aaRedirectToInTouch IOServerKey, InTouchNodeKey
```

Arguments
IOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with no default.

InTouchNodeKey
The unique numerical identifier of the named InTouch node. This value is of data type int, with no default.

Remarks
When you redirect to InTouch HMI software, all tag values will come from the HMI, not directly from the I/O Server. If you redirect an I/O Server, all topics and tags for that particular I/O Server are affected.

Permission
Execute permission defaults to the **aaAdministrators** group.

aaSetAISamples

Sets the number of samples that the active image can hold for a specified tag. This stored procedure is used by the Wonderware Historian and should not be executed by users.

Syntax

```
aaSetAISamples TagName, Samples
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system.This value is of data type nvarchar(256), with no default.

Samples
The number of samples that the active image holds for the tag. 0 indicates that the active image is using the default of 65 values. The higher the number of samples, the higher the load on memory resources. This value is of data type int, with no default.

Remarks

The system initially sets the number of samples for each tag to 65. The number of samples for a tag is stored in the **SampleInActiveImage** column of the **Tag** table.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaSetCalculatedAISamples

Updates the value of the SamplesInActiveImage column of the Tag table with the current value of the CalculatedAISamples column.

Syntax

```
aaSetCalculatedAISamples
```

Remarks

This stored procedure executes at system startup. Also, this stored procedure only executes if the AIAutoResize system parameter is set to 1.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaSetServerTimeStamp

Specifies whether or not incoming data values will be timestamped using the time of the local Wonderware Historian.

Syntax

```
aaSetServerTimeStamp TopicName, ServerTimeStamp
```

Arguments

TopicName
The name of the topic. This value is of data type nvarchar(80), with no default.

ServerTimeStamp
Used to specify whether local timestamping by the Wonderware Historian is used. 0 = The IDAS timestamp is used; 1 = The Wonderware Historian time is used for the timestamp. If a fast-changing tag is configured to use server timestamping, the packet of data that is sent to the storage subsystem may contain multiple data values with the same timestamp, which may affect data calculations, such as for swinging door storage. This value is of data type bit, with a default of 0.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaSetStorageRule

Sets storage rules at various levels of the tag definition.

Syntax

```
aaSetStorageRule Type, Key, StorageType, StorageRate,
  TimeDB, ValueDB, AcqType, DBType, RateDB,
  ServerTimeStamp, LateData, IdleDuration, ProcInterval
```

Arguments

Type
The level at which the new storage rule will be set for the tag definition. 1 = I/O Server; 2 = Topic. This value is of data type tinyint, with no default.

Key
The database key value for the relevant type, either the I/O Server key or the topic key. This value is of data type int, with no default.

StorageType
The type of storage defined for the tag. 0 = Not stored; 1 = Cyclic; 2 = Delta; 3 = Forced storage; 17 = The storage type has been changed from cyclic to "not stored." 18 = The storage type has been changed from delta to "not stored." 19 = The storage type has been changed from "forced" to "not stored." This value is of data type tinyint, with no default.

StorageRate
The rate at which the tag is stored if the storage type is cyclic. The rate is in seconds. This value is of data type int, with a default of 0.

TimeDB
The minimum time, in milliseconds, between stored values for a single tag. Any value changes that occur within the time deadband are not stored. The time deadband applies to delta storage only. A time deadband of 0 indicates that the system will store the value of the tag each time it changes. This value is of data type int, with a default of -1.

ValueDB
The percentage of the difference between the minimum and maximum engineering units for the tag. Any data values that change less than the specified deadband are not stored. The value deadband applies to delta storage only. A value of 0 indicates that a value deadband will not be applied. This value is of data type float, with a default of -1.

AcqType
Used to turn acquisition on or off. 0 = Acquisition off; 1 = Acquisition on. This value is of data type smallint, with a default of -1.

DBType
The type of delta storage deadband to be applied for the tag. This setting is only in effect if delta storage is configured for the tag. 1= Time and/or value deadband; 2 = Rate (swinging door) deadband. This value is of data type smallint, with a default of -1.

RateDB
Used to percentage of deviation in the full-scale value range for an analog tag. The swinging door (rate) deadband applies to delta storage only. Time and/or value deadbands can be used in addition to the swinging door deadband. Any value greater than 0 can be used for the deadband. A value of 0 indicates that a swinging door deadband will not be applied. This value is of data type float, with a default of -1.

ServerTimeStamp
Used to specify whether local timestamping by the Wonderware Historian is used. 0 = The IDAS timestamp is used; 1 = The Wonderware Historian time is used for the timestamp. If a fast-changing tag is configured to use server timestamping, the packet of data that is sent to the storage subsystem may contain multiple data values with the same timestamp, which may affect data calculations, such as for swinging door storage. This value is of data type smallint, with a default of -1.

LateData
Used to enable acquisition of "late" data. 0 = Late data disabled; 1 = Late data enabled. This value is of data type smallint, with a default of -1.

IdleDuration
The amount of time, in seconds, before data is processed from the I/O Server. For example, if you set this value to 60 seconds, data from this I/O Server is cached and only processed by the storage engine after no more data has been received from the I/O Server for at least 60 seconds. This value is of data type int, with a default of 60.

ProcInterval
The amount of time, in seconds, after which late data from the I/O Server is processed, regardless of the idle duration. If the nature of the data is such that the idle duration is never satisfied, the historian storage engine processes data from the topic at least one time every processing interval. The processing interval defaults to twice the idle duration and cannot be set to a value less than the idle duration. This value is of data type int, with a default of 120.

Remarks
To ignore an argument, set the value to -1.

Permission
Execute permission defaults to the **aaAdministrators** and **aaPowerUsers** groups.

aaSetTagStorage
Sets storage on or off from various level of the tag definition.

Syntax
```
aaSetTagStorage Type, List, Set
```

Arguments
Type
The level at which the new storage rule will be set for the tag definition. 1 = I/O Server; 2 = Topic. This value is of data type tinyint, with no default.

List
If the type is an I/O Server, topic or public group, the IDENTITY key(s) of the relevant type. If the type is a tag, a list of tagnames separated by commas. This value is of data type nvarchar(4000), with no default.

Set
Used to set storage on or off. Valid values are ON, OFF. This value is of dataype varchar(3), with no default.

Remarks
This stored procedure applies to analog, discrete, string and complex tag types.

Permissions
Execute permission defaults to the **aaAdministrators** and **aaPowerUsers** groups.

Examples
The following example turns data storage off for all tags associated with I/O Servers that are identified by the IOServerKeys 2 and 3.

```
aaSetTagStorage 1, '2,3', 'OFF'
```

The following example turns data storage on for the listed tags.

```
aaSetTagStorage 3, 'Tag1, Tag2, Tag3', 'ON'
```

aaSnapshotDetailSelect

Returns snapshot information from the columns of the **SnapshotDetail** table, based on the storage size.

Syntax
```
aaSnapshotDetailSelect StorageSize
```

Arguments
StorageSize
The storage size, in bytes, of the tag value: -1 = Blob; 0 = Variable length string; 1 = 1 byte; 2 = 2 byte; 4 = 4 byte; 8 = 8 byte. This value is of data type int, with a default of NULL.

Remarks
If you do not pass an argument for the storage size, information for all storage sizes in the table will be returned.

Permission
Execute permission defaults to the **public** group.

aaSnapshotDetailUpdate

Updates the **SnapshotDetail** table.

Syntax

```
aaSnapshotDetailUpdate StorageSize, SnapshotSize,
  ImageTime, ThresholdTime
```

Arguments

StorageSize
The storage size, in bytes, of the tag value: -1 = Blob; 0 = Variable length string; 1 = 1 byte; 2 = 2 byte; 4 = 4 byte; 8 = 8 byte. This value is of data type int, with no default.

SnapshotSize
The maximum size of the snapshot, in bytes. If this limit is reached, a new snapshot is created. This value is of data type int, with no default.

ImageTime
The interval, in seconds, between updates to the snapshot file. The snapshot file is updated with tag value information from the snapshot buffer, which resides in memory. This value is of data type int, with no default.

ThresholdTime
The maximum amount of time, in seconds, that can elapse before a new snapshot is automatically created, provided that the value for the snapshot size has not been reached. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaSnapToSummary

Used by the event system when configuring summary operations.

Syntax

```
aaSnapToSummary OpKey, Start, End, DateStamp
```

Arguments

OpKey
An internal variable that identifies the summary operation to perform. This value is of data type int, with no default.

Start
The starting timestamp for the calculation. This value is of data type datetime, with no default.

End
The ending timestamp for the calculation. This value is of data type datetime, with no default.

DateStamp
The time the summary operation was performed. This value is of data type smalldatetime, with no default.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaSpaceManager

Used by the system to manage the amount of disk space used to store historical data for summaries and events.

Syntax
```
aaSpaceManager
```

Remarks
This stored procedure is automatically run by the system every ten minutes. This stored procedure executes the aaDeleteOlderEvents and aaDelectOlderSummaries stored procedures to clear out old historical data. The duration for which event and summary history is kept is based on system parameters stored in the SystemParameter table.

Permission
Execute permission defaults to the **aaAdministrators** group.

aaStorageLocationSelect

Selects a storage location.

Syntax
```
aaStorageLocationSelect StorageType, StorageNodeKey
```

Arguments
StorageType
The type of storage used for the specified location. 1 = Circular; 2 = Alternate; 3 = Buffer; 4 = Permanent. There can be only one storage location of each type. This value is of data type int, with a default of NULL.

StorageNodeKey
The unique numerical identifier for the storage node. This value is of data type int, with a default of 1.

Permission
Execute permission defaults to the **public** group.

aaStorageLocationUpdate

Updates the storage location.

Syntax

```
aaStorageLocationUpdate StorageType, StorageNodeKey,
  SortOrder, Path, MaxMBSize, MinMBThreshold
```

Arguments

StorageType

The type of storage used for the specified location. 1 = Circular; 2 = Alternate; 3 = Buffer; 4 = Permanent. There can be only one storage location of each type. This value is of data type int, with no default.

StorageNodeKey

The unique numerical identifier for the storage node. This value is of data type int, with no default.

SortOrder

Applies only to the alternate area. If more than one location is defined, the sort order determines the order in which the alternate areas are used. Reserved for future use. This value is of data type int, with no default.

Path

The path to the storage location. The circular storage location must be a local drive on the server machine, and the path must be specified using normal drive letter notation (for example, c:\Historian\Data\Circular). For a tier-1 historian, the alternate, buffer, and permanent storage locations can be anywhere on the network. For a tier-2 historian, the buffer and permanent storage locations can be anywhere on the network, but the alternate storage location must be on a local drive. The ArchestrA service user must have full access to network locations. The locations must be specified using UNC notation. Mapped drives are not supported. This value is of data type nvarchar(255), with no default.

MaxMBSize

The limit, in megabytes, for the amount of data to be stored to the specified location. The maximum size applies to circular and alternate storage only. If the maximum size is set to 0, all available space at the storage location is used. This value is of data type int, with no default.

MinMBThreshold
The minimum amount of disk space, in megabytes, at which the system attempts to start freeing up space. The threshold applies to circular and alternate storage only. Typically, you should multiply the size of the average history block (before any compression) by 1.5 to determine the minimum threshold. This value is of data type int, with no default.

MaxAgeThreshold
The age, in days, of data that will be deleted by system to free up disk space. The threshold applies to circular and alternate storage only. The minimum age is 2 days. A value of 0 indicates that no age threshold is applied. This value is of data type int, with a default of 0.

Permission
Execute permission defaults to the **aaAdministrators** group.

aaStringDetail

Returns the details for a specified string tag.

Syntax

```
aaStringDetail TagList
```

Arguments

TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

Permission
Execute permission defaults to the **public** group.

aaStringTagDelete

Deletes a string tag.

Syntax

```
aaStringTagDelete wwTagKey
```

Arguments

wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with a default of NULL.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaStringTagInsert

Inserts a string tag.

Syntax

```
aaStringTagInsert TagName, Description,
  AcquisitionType, StorageType, StorageRate, ItemName,
  TimeDeadband, CreatedBy, DateCreated, MaxLength,
  InitialValue, TopicKey, IOServerKey, CurrentEditor,
  DoubleByte, SamplesInActiveImage, ServerTimeStamp,
  DeadbandType
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

Description
The description of the tag. This value is of data type nvarchar(512), with a default of an empty string.

AcquisitionType
The method by which the tag's value is acquired. If the tag value is acquired from an I/O Server, specify the name of the I/O Server, topic, and item. 0 = Not acquired; 1 = Acquired via an I/O Server; 2 = Acquired via MDAS or a manual update; 3 = System driver. This value is of data type tinyint, with a default of 1.

StorageType
The type of storage defined for the tag. 0 = Not stored; 1 = Cyclic; 2 = Delta; 3 = Forced storage; 17 = The storage type has been changed from cyclic to "not stored." 18 = The storage type has been changed from delta to "not stored." 19 = The storage type has been changed from "forced" to "not stored." This value is of data type smallint, with a default of 2.

StorageRate
The rate at which the tag is stored if the storage type is cyclic. The rate is in milliseconds. This value is of data type int, with a default of 1000.

ItemName
The address string of the tag. This value is of data type nvarchar(256), with a default of an empty string.

TimeDeadband
The minimum time, in milliseconds, between stored values for a single tag. Any value changes that occur within the time deadband are not stored. The time deadband applies to delta storage only. A time deadband of 0 indicates that the system will store the value of the tag each time it changes. This value is of data type int, with a default of 0.

CreatedBy
The name of the user or application that created the tag. This value is of data type nvarchar(256), with a default of an empty string.

DateCreated
The date that the tag was created. This value is of data type datetime, with a default of NULL.

MaxLength
The maximum number of characters for the string. This value is of data type smallint, with a default of 131. Valid values are: 8, 16, 24, 32, 48, 64, 128, 131, 256, 512.

InitialValue
The initial value as imported from an external source (for example, from InTouch). This value is of data type nvarchar(512), with a default of an empty string.

TopicKey
The unique numerical identifier for the topic. This value is of data type int, with a default of NULL.

IOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with a default of NULL.

CurrentEditor
Used to specify which application or editing environment controls the tag definition. Tags imported from the InTouch HMI software use InTouch as the current editor. If modifications are made to an imported tag in the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. If a re-import is performed, any modifications made using the Configuration Editor are preserved. You can manually maintain InTouch as the current editor for re-importing; however, all changes made to the tag using the Configuration Editor are lost during the re-import. Tags (attributes) that are initially configured using Wonderware Application Server use the ArchestrA Integrated Development Environment (IDE) as the current editor. If you modify an Application Server tag using the historian Configuration Editor, then the current editor for the tag is changed to Wonderware Historian. However, the next time you redeploy the engine, the changes are not preserved. 0 = Wonderware Historian; 1 = InTouch; 2 = Wonderware Application Server. This value is of data type int, with a default of 0.

DoubleByte
Used to specify whether or not to store the string as a double-byte string. 0 = Not stored as double-byte; 1 = Stored as double-byte. This value is of data type tinyint, with a default of 0.

SamplesInActiveImage
The number of samples that the active image holds for the tag. 0 indicates that the active image is using the default of 65 values. The higher the number of samples, the higher the load on memory resources. This value is of data type int, with a default of 0.

ServerTimeStamp
Used to specify whether local timestamping by the Wonderware Historian is used. 0 = The IDAS timestamp is used; 1 = The Wonderware Historian time is used for the timestamp. If a fast-changing tag is configured to use server timestamping, the packet of data that is sent to the storage subsystem may contain multiple data values with the same timestamp, which may affect data calculations, such as for swinging door storage. This value is of data type bit, with a default of 0.

DeadbandType
The type of delta storage deadband to be applied for the tag. This setting is only in effect if delta storage is configured for the tag. 1= Time and/or value deadband; 2 = Rate (swinging door) deadband. This value is of data type smallint, with a default of 1.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaStringTagSelect
Selects a string tag.

Syntax
```
aaStringTagSelect wwTagKey
```

Arguments
wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with a default of NULL.

Permission
Execute permission defaults to the **public** group.

aaStringTagUpdate
Updates a string tag.

Syntax
```
aaStringTagUpdate wwTagKey, TagName, Description,
  AcquisitionType, StorageType, StorageRate, ItemName,
  TimeDeadband, CreatedBy, DateCreated, MaxLength,
  InitialValue, TopicKey, IOServerKey, CurrentEditor,
  DoubleByte, SamplesInActiveImage, ServerTimeStamp,
  DeadbandType
```

Arguments
wwTagKey
The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

The remaining arguments are the same as for the **aaStringTagInsert** stored procedure. However, only AcquisitionType, StorageType, CreatedBy, DateCreated, MaxLength, DoubleByte, SamplesInActiveImage, ServerTimeStamp, and DeadbandType have defaults.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaSummaryActionInsert

Used by the event subsystem to perform a summary operation for the specified tag.

Syntax

```
aaSummaryActionInsert EventTagName, SumDateTime
```

Arguments

EventTagName
The name of the event tag with which the summary operation is associated. This value is of data type nvarchar(256), with no default.

SumDateTime
The timestamp to use when storing the result of the calculation. The timestamp can be either the time when the calculation period starts or ends. This value is of data type datetime, with no default.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaSummaryDetail

Returns summary details for one or more tags that are associated with a particular summary operation. The type of aggregation for the tag can optionally be included for each tag that you list.

The starting and ending times are used to specify the time at which the calculation started/ended for the operation.

Syntax

```
aaSummaryDetail TagList, StartTime, EndTime, OrderBy
```

Arguments

TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

StartTime
The starting timestamp for the calculation. This value is of data type nvarchar(50), with no default.

EndTime
The ending timestamp for the calculation. This value is of data type nvarchar(50), with no default.

OrderBy
The column by which the results will be ordered. By default, the TagName column is used. This value is of data type nvarchar(500).

Permission

Execute permission defaults to the **public** group.

Examples

This example returns the average and miniumum values for 'ReactTemp' and the maximum value for 'ReactLevel' between 12:12 p.m. and 2:14 p.m. on May 12, 2001. The returned rows are ordered by the date of the summary.

```
aaSummaryDetail "ReactTemp('AVG','MIN'),
  ReactLevel('MAX')", "2001-05-12 12:12:00.000",
  "2001-05-12 12:14:00.000", "SummaryDate"
```

This example returns all aggregate values for 'ReactTemp' and 'ReactLevel' between 12:12 p.m. and 2:14 p.m. on May 12, 2001.

```
aaSummaryDetail "ReactTemp, ReactLevel", "2001-05-12
  12:12:00.000", "2001-05-12 12:14:00.000"
```

aaSummaryOperationDelete

Deletes a summary operation.

Syntax

```
aaSummaryOperationDelete OperationKey
```

Arguments

OperationKey
The unique numerical identifier for the summary operation. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaSummaryOperationInsert

Inserts a summary operation that will be associated with the specified event tag.

Syntax

```
aaSummaryOperationInsert TagName, CalcType, Duration,
  Resolution, TimeStamp, Description
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system.This value is of data type nvarchar(256), with no default.

CalcType
The type of calculation to be performed: SUM, MAX, MIN, or AVG. This value is of datatype char(3), with no default.

Duration
The period, in seconds, for which the calculation is performed. This value is of data type real, with no default.

Resolution
The sampling rate, in milliseconds, for retrieving the data in cyclic mode. The system returns values stored over the requested time period at the interval specified by the resolution. For example, if you specify a 5000 ms resolution, the system queries for all data during the time period and then only returns those values that occur at each 5000 ms interval, starting with the start date and ending with the end date. This value is of data type int, with no default.

TimeStamp
The timestamp to use when storing the result of the calculation. The timestamp can be either the time when the calculation period starts or ends. 0 = Beginning of the calculation period; 1 = End of the calculation period. This value is of data type tinyint, with no default.

Description
The description of the summary operation. This value is of data type nvarchar(50), with a default of NULL.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaSummaryOperationSelect

Selects a summary operation.

Syntax

```
aaSummaryOperationSelect EventTagName, CalcType,
  Duration, Resolution, TimeStamp
```

Arguments

EventTagName
The name of the event tag with which the summary operation is associated. This value is of data type nvarchar(256), with a default of NULL.

The remaining arguments are the same as for the **aaSummaryOperationInsert** stored procedure. However, all of the arguments have a default of NULL.

Remarks

The arguments of this stored procedure are used in three ways: (1) if no arguments are specified, all summary operations will be returned; (2) if the EventTagName argument is specified, all summary operations for that event tag will be returned; (3) if all arguments are specified, only the summary operation that matches the criteria will be returned.

Permission

Execute permission defaults to the **public** group.

aaSummaryOperationUpdate

Updates the summary operation that is associated with a specified event tag.

Syntax

```
aaSummaryOperationUpdate OperationKey, TagName,
  CalcType, Duration, Resolution, TimeStamp,
  Description
```

Arguments

OperationKey
The unique numerical identifier for the summary operation. This value is of data type int, with no default.

The remaining arguments are the same as for the **aaSummaryOperationInsert** stored procedure. However, only the Description argument has a default.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaSummaryTagListDelete

Deletes summary information for a tag.

Syntax

```
aaSummaryTagListDelete SumVarKey
```

Arguments

SumVarKey
The unique numerical identifier for a summarized tag. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaSummaryTagListInsert

Inserts summary information for a specified tag.

Syntax

```
aaSummaryTagListInsert TagName, OperationKey,
  LowerLimit, UpperLimit, Description
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

OperationKey
The unique numerical identifier for the summary operation. This value is of data type int, with no default.

LowerLimit
The lower limit of validity for the tag's value. Values lower than this limit are not used in the calculation. By default, this value is set to -1000000000. This value is of data type real, with a default of NULL.

UpperLimit
The upper limit of validity for the tag's value. Values higher than this limit are not used in the calculation. By default, this value is set to 1000000000. This value is of data type real, with a default of NULL.

Description
The description of the summarized tag. This normally describes the result of the operation, although this description can be the same as that of the tag on which the operation is performed. This value is of data type nvarchar(50), with a default of NULL.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaSummaryTagListSelect

Selects summary information for a tag.

Syntax

```
aaSummaryTagListSelect OperationKey, TagName
```

Arguments

OperationKey
The unique numerical identifier for the summary operation. This value is of data type int, with no default.

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaSummaryTagListUpdate

Updates summary information for a specified tag.

Syntax

```
aaSummaryTagListUpdate SumVarKey, TagName,
  OperationKey, LowerLimit, UpperLimit, Description
```

Arguments

SumVarKey
The unique numerical identifier for a summarized tag. This value is of data type int, with no default.

The remaining arguments are the same as for the **aaSummaryTagListInsert** stored procedure.

Permission

Execute permission defaults to the **aaAdministrators** group.

aaSystemConfigNSExpand

Expands the tree view under a single object in the system namespace. This stored procedure is used by the System Management Console.

Syntax

```
aaSystemNSExpand PKey, FKey1, FKey2, FKey3, TokenType,
  FilterStr
```

Arguments

PKey
A local variable used to identify the object in the namespace. This value is of data type int, with no default.

FKey1-FKey3
A local variable used to determine the position of the object in the tree view. This value is of data type int, with no default.

TokenType
The type of system namespace group. 1000010 = Data Acquisition; 1000017 = System Driver; 1000018 = IDASs; 1000019 = I/O Servers. This value is of data type int, with no default.

FilterStr
Used to order the tagnames in the namespace. This value is of data type nvarchar(100), with a default of NULL.

Permission
Execute permission defaults to the **public** group.

aaSystemNSExpand
Expands the tree view under a single object in the system namespace.

Syntax
```
aaSystemNSExpand PKey, FKey1, FKey2, FKey3, TokenType,
  FilterStr
```

Arguments
PKey
A local variable used to identify the object in the namespace. This value is of data type int, with no default.

FKey1-FKey3
A local variable used to determine the position of the object in the tree view. This value is of data type int, with no default.

TokenType
The type of system namespace group. 1000010 = Data Acquisition; 1000017 = System Driver; 1000018 = IDASs; 1000019 = I/O Servers. This value is of data type int, with no default.

FilterStr
Used to order the tagnames in the namespace. This value is of data type nvarchar(100), with a default of NULL.

Permission
Execute permission defaults to the **public** group.

aaSystemNSExpand2

Expands the tree view under a single object in the system namespace.

Note This stored procedure is a simpler version of the **aaSystemNSExpand** stored procedure.

Syntax

```
aaSystemNSExpand2 PKey, FilterStr
```

Arguments

PKey
A local variable used to identify the object in the namespace. This value is of data type int, with no default.

FilterStr
Used to order the tagnames in the namespace. This value is of data type nvarchar(100), with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaSystemParameterSelect

Returns details for a specified system parameter, such as a description of the parameter, the current value, and so on. If you do not specify a name, the stored procedure returns details for all defined system parameters.

Syntax

```
aaSystemParameterSelect name
```

Arguments

Name
The unique name for the system parameter. This value is of data type nvarchar(50), with a default of NULL.

Permission

Execute permission defaults to the **public** group.

aaSystemParameterUpdate

Updates the value and description for a specified system parameter. If you do not provide a description, the previous description is used.

Syntax

```
aaSystemParameterUpdate Name, Value, Description
```

Arguments

Name
The unique name for the system parameter. This value is of data type nvarchar(50), with no default.

Value
The value of the system parameter. This value is of data type sql_variant, with no default.

Description
The description of the system parameter. This value is nvarchar(255), with a default of NULL.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaTagConfig

Used by the internal Configuration Manager when the Wonderware Historian starts.

Syntax

```
aaTagConfig
```

Remarks

This stored procedure takes a snapshot of the system configuration at the tag level.

Permission

Execute permission defaults to the **public** group.

aaTagConfigModified

Used by the internal configuration object.

Syntax

```
aaTagConfigModified
```

Remarks

This stored procedure has the same functionality as the **aaTagConfig** stored procedure, but only retrieves the database modifications pending when a commit of changes is performed.

Permission

Execute permission defaults to the **public** group.

aaTagConfigSelect

Used by the System Management Console to return a list of tags associated with a particular engineering unit (for analog tags) or message (for discrete tags).

Syntax

```
aaTagConfigSelect, TagType, Key, FilterStr
```

Arguments

TagType
The type of tag to retrieve. 1 = Analog; 2 = Discrete. This value is of data type int, with no default.

Key
The database key value for the relevant type, either the message key or the engineering unit key. This value is of data type int, with no default.

FilterStr
Used to order the tagnames in the namespace. This value is of data type nvarchar(100), with a default of N%.

Permission

Execute permission defaults to **public** group.

aaTagInfo

Returns definition information for each specified tag.

Syntax

```
aaTagInfo TagList
```

Arguments

TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

Permission

Execute permission defaults to the **public** group.

aaTagType

Returns the tag type for each specified tag.

Syntax

```
aaTagType TagList
```

Arguments

TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

Permission

Execute permission defaults to the **public** group.

aaTimeDetectorDetailInsert

Inserts time detector details that are associated with a specified event tag.

Syntax

```
aaTimeDetectorDetailInsert FrequencyID, TagName,
  Periodicity, StartDateTime, RunTimeDay, RunTimeHour,
  RunTimeMin
```

Arguments

FrequencyID
The unique numerical identifier for the frequency. Used to link a frequency with a time-based detector. 1= Hourly; 2 = Daily; 3 = Weekly; 4 = Monthly; 5 = Periodic; 6 = Other (Reserved for future use). This value is of data type int, with no default.

TagName
The unique name of the tag within the Wonderware Historian system. This value is of data type nvarchar(256), with no default.

Periodicity
The interval period in minutes between detector events. Only used for a periodic detection. This value is of data type int, with no default.

StartDateTime
The timestamp from which the time detector starts. Only used for a periodic detection. This value is of data type datetime, with no default.

RunTimeDay
In the context of a weekly detector, RunTimeDay maps the week day number (0 = Sunday – 6 = Saturday). In the context of a monthly detector, RunTimeDay maps to the day of the month. Not used for periodic detections. This value is of data type tinyint, with no default.

RunTimeHour
The hour of the day at which the time detector triggers. Not used for periodic detections. This value is of data type tinyint, with no default.

RunTimeMin
The minute of the hour at which the time detector triggers. Not used for periodic detections. This value is of data type tinyint, with no default.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaTimeDetectorDetailSelect

Selects the time detector from the **TimeDetectorDetail** table that is associated with the specified event tag.

Syntax

```
aaTimeDetectorDetailSelect TagName
```

Arguments

TagName
The unique name of the tag within the Wonderware Historian system.This value is of data type nvarchar(256), with a default of NULL.

Permission
Execute permission defaults to the **public** group.

aaTimeDetectorDetailUpdate

Updates the time detector in the **TimeDetectorDetail** table that is associated with the specified event tag.

Syntax

```
aaTimeDetectorDetailUpdate FrequencyID, TagName,
  Periodicity, StartDateTime, RunTimeDay, RunTimeHour,
  RunTimeMin
```

Arguments
The arguments are the same as for the **aaTimeDetectorDetailUpdate** stored procedure. However, none of the arguments have defaults.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaTopicDelete

Deletes an I/O topic.

Syntax

```
aaTopicDelete TopicKey
```

Arguments

TopicKey
The unique numerical identifier for the topic. This value is of data type int, with no default.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaTopicInsert

Inserts an I/O topic.

Syntax

```
aaTopicInsert StorageNodeKey, IOServerKey, Name,
  TimeOut, LateData, IdleDuration, ProcessingInterval
```

Arguments

StorageNodeKey
The unique numerical identifier for the storage node. This value is of data type int, with no default.

IOServerKey
The unique numerical identifier for the I/O Server. This value is of data type int, with no default.

Name
The name of the topic. This value is of data type nvarchar(80), with no default.

TimeOut
The time span, in milliseconds, in which a data point must be received on the topic. If no data point is received in this time span, the topic is considered "dead." The historian disconnects and then attempts to reconnect to the topic. This value is of data type int, with a default of 60000.

LateData
Used to enable acquisition of "late" data. 0 = Late data disabled; 1 = Late data enabled. This value is of data type bit, with a default of 0.

IdleDuration
The amount of time, in seconds, before data is processed from the I/O Server. For example, if you set this value to 60 seconds, data from this I/O Server is cached and only processed by the storage engine after no more data has been received from the I/O Server for at least 60 seconds. This value is of data type int, with a default of 60.

ProcessingInterval
The amount of time, in seconds, after which late data from the I/O Server is processed, regardless of the idle duration. If the nature of the data is such that the idle duration is never satisfied, the historian storage engine processes data from the topic at least one time every processing interval. The processing interval defaults to twice the idle duration and cannot be set to a value less than the idle duration. This value is of data type int, with a default of 120.

Permission
Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaTopicSelect

Selects an I/O topic.

Syntax
```
aaTopicSelect TopicKey
```

Arguments
TopicKey
The unique numerical identifier for the topic. This value is of data type int, with a default of NULL.

Permission
Execute permission defaults to the **public** group.

aaTopicUpdate

Updates an I/O topic.

Syntax
```
aaTopicUpdate TopicKey, StorageNodeKey, IOServerKey,
  Name, TimeOut, LateData, IdleDuration,
  ProcessingInterval
```

Arguments
TopicKey
The unique numerical identifier for the topic. This value is of data type int, with no default.

The remaining arguments are the same as for the **aaTopicInsert** stored procedure. However, only the TimeOut, LateData, IdleDuration, and ProcessingInterval arguments have defaults.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaUpdateCalculatedAISamples

Used by the system to write the optimum number of samples in the active image to the CalculatedAISamples column in the Tag table. This stored procedure is used by the Wonderware Historian and should not be executed by users.

Syntax

```
aaSetCalculatedAISamples TagKey, Samples
```

TagKey

The unique numerical identifier of a tag within a single Wonderware Historian. This value is of data type int, with no default.

Samples

The number of samples that the active image holds for the tag. 0 indicates that the active image is using the default of 65 values. The higher the number of samples, the higher the load on memory resources. This value is of data type int, with no default.

Remarks

If the AIAutoResize system parameter is set to 1, the system continuously recalculates the optimum number of samples for each tag based on the data rates received. If the calculated value exceeds the current value in the database, then the system updates the **CalculatedAISamples** column in the **Tag** table.

Permission

Execute permission defaults to the **aaPowerUsers** and **aaAdministrators** groups.

aaUserAccessLevelSelect

Returns the access level associated with the currently logged on user. Access levels are: 1 = Undefined Wonderware Historian user (for example, **public**); 2 = User (**hUser** permissions); 3 = **PowerUser** (**hPowerUser** permission); 3 = Admin (**hAdmin** permissions); 9999 = dbo.

Syntax

```
aaUserAccessLevelSelect
```

Remarks

The access level values correspond to values in the **UserDetail** table, which is populated during installation.

Permission

Execute permission defaults to the **public** group.

aaUserDetailUpdate

Allows the **UserDetail** table to be populated from information contained in the **sysusers** table.

Syntax

```
aaUserDetailUpdate
```

Permission

Execute permission defaults to the **aaAdministrators** group.

Extended Stored Procedures

An extended stored procedure can trigger an action outside of SQL Server, the results of which can be stored back into the SQL Server. All of the extended stored procedures for the Wonderware Historian are contained in one single dynamic-link library, called aahSQLXP.dll, and communicate with a historian as MDAS clients. For example, to query data from the AnalogHistory table, the xp_AnalogHistory extended stored procedure will make a call to the aahSQLXP.dll. This .dll will perform the actual data query on the AnalogHistory table (which is only supported by the historian) and return the results to the Microsoft SQL Server.

Extended stored procedures must be executed from the **master** database, or must be qualified with a master prefix if executed from the Runtime database. For example, `EXEC master..xp_ProcList`. Also, these extended stored procedures are case-sensitive, if the SQL Server was installed in case-sensitive mode. Execute permission for most of these extended stored procedures defaults to the **public** group. Extended stored procedures that change system settings or perform administrative actions may only be executed by members of the aaAdministrators group.

Some of the historian extended stored procedures allow high-level retrieval access to archived history data through Microsoft SQL Server queries. Other stored procedures also provide a set of utility functions to work with historian history blocks and deadbands.

History Extended Stored Procedures

The history extended stored procedures allow you to query historical data from within stored procedures created on the Microsoft SQL Server. Any normal SQL Server function that can call a stored procedure can call any of these extended stored procedures.

The data values returned will be the most current values that exist in the database. The ability to select the version of history (latest or original) is not supported. Use the Wonderware Historian OLE DB provider to access versioned data values.

For descriptions of the parameters used by the extended stored procedures, see "Extended Stored Procedure Arguments" on page 252.

xp_AnalogHistory

Returns data from the AnalogHistory table.

Syntax

```
xp_AnalogHistory StartTime, EndTime, Resolution, Tag1
  [,Tag2,...]
```

xp_AnalogHistoryDelta

Returns a row from the AnalogHistory table for each instance where a stored analog value changed (delta retrieval).

Syntax

```
xp_AnalogHistoryDelta StartTime, EndTime, MaxRowCount,
  Tag1 [,Tag2,...]
```

xp_AnalogWideHistory

Returns data from the AnalogWideHistory table. Each row returned contains a value in each column for each tag's value at a specified time.

Syntax

```
xp_AnalogWideHistory StartTime, EndTime, Resolution,
  Tag1 [,Tag2,...]
```

xp_AnalogWideHistoryDelta

Returns data from the AnalogWideHistory table for each instance where a stored analog value changed (delta retrieval). Each row returned contains a value in each column for each tag's value at a specified time.

Syntax

```
xp_AnalogWideHistoryDelta StartTime, EndTime,
  MaxRowCount, Tag1 [,Tag2,...]
```

xp_DiscreteHistory

Returns data from the DiscreteHistory table.

Syntax

```
xp_DiscreteHistory StartTime, EndTime, Resolution, Tag1
  [,Tag2,...]
```

xp_DiscreteHistoryDelta

Returns a row from the DiscreteHistory table for each instance where a stored discrete value changed (delta retrieval).

Syntax

```
xp_DiscreteHistoryDelta StartTime, EndTime,
  MaxRowCount, Tag1 [,Tag2,...]
```

xp_DiscreteWideHistory

Returns data from the DiscreteWideHistory table. Each row returned contains a value in each column for each tag's value at a specified time.

Syntax

```
xp_DiscreteWideHistory StartTime, EndTime, Resolution,
  Tag1 [,Tag2,...]
```

xp_DiscreteWideHistoryDelta

Returns data from the DiscreteWideHistory table for each instance where a stored discrete value changed (delta retrieval). Each row returned contains a value in each column for each tag's value at a specified time.

Syntax

```
xp_DiscreteWideHistoryDelta StartTime, EndTime,
  MaxRowCount, Tag1 [,Tag2,...]
```

Utility Extended Stored Procedures

Utility extended stored procedures provided with the Wonderware Historian are:

- xp_DiskCopy
- xp_NewHistoryBlock
- xp_ProcList
- xp_RescanHistoryBlocks
- xp_SetStorageTimeDeadband
- xp_SetStorageValueDeadband
- xp_SetStoreForwardEvent

xp_DiskCopy

Copies history blocks or a subset of history blocks to the "archive" historical path (permanent path).

Syntax

```
xp_DiskCopy StartTime, EndTime, Description
```

Parameters

For more information on the parameters, see "Extended Stored Procedure Arguments" on page 252.

Remarks

The **xp_DiskCopy** extended stored procedure will not allow a history block to be copied if the time range specified is already archived in permanent storage. You must first delete the archived block(s) that falls within the specified time range before performing the block copy. This prevents overwriting the archived block with a newer block that has had old data inserted with the same timestamps as the already archived data.

Example

For example:

```
xp_DiskCopy "20010220 13:10:00:000", "20010220
  14:00:00:000", "Plant trip 02/20/2001"
```

xp_NewHistoryBlock

Stops data from being written to the current history block and starts a new one. This process will take approximately 10 minutes, but no data is lost. The system acts the same as if it were a scheduled block changeover.

Syntax

```
xp_NewHistoryBlock
```

Remarks

When this extended stored procedure is executed, a message will be logged to the message log file as for a regular block changeover. However, if executed manually in SQL Server Query Analyzer, a message will also appear in the Results section if the execution was successful. To verify that the block changeover actually occurred, you must still check in the System Management Console message log. For more information, see "Monitoring System Messages" in Chapter 10, "Monitoring the System," in your *Wonderware Historian Administration Guide*.

xp_ProcList

Returns syntax details for all of the extended stored procedures for the Wonderware Historian that exist on the Microsoft SQL Server.

Syntax

```
xp_ProcList
```

xp_RescanHistoryBlocks

Refreshes the system with any history block changes since the last scan.

Syntax

```
xp_RescanHistoryBlocks
```

xp_SetStorageTimeDeadband

Sets the storage time deadband for one or more tags while the Wonderware Historian is running.

Syntax

```
xp_SetStorageTimeDeadband <TimeDeadband>, <Tag1> [,
  <Tag2>,...]
```

Arguments

For more information on the parameters, see "Extended Stored Procedure Arguments" on page 252.

Remarks

The **xp_SetStorageTimeDeadband** extended stored procedure is a server function that will modify the delta storage rule for the specified tag. It does not update the database and is valid until the server is restarted.

Example

This example sets the storage time deadband to 2000 milliseconds for the analog tag "BoilerTag:"

```
xp_SetStorageTimeDeadband 2000, 'BoilerTag'
```

xp_SetStorageValueDeadband

Sets the storage value deadband for one or more tags while the Wonderware Historian is running.

Syntax

```
xp_SetStorageValueDeadband <ValueDeadband>, <Tag1> [,
  <Tag2>,...]
```

Arguments

For more information on the parameters, see "Extended Stored Procedure Arguments" on page 252.

Remarks

The **xp_SetStorageValueDeadband** extended stored procedure is a server function that will modify the delta storage rule for the specified tag. It does not update the database and is valid until the server is restarted.

Example

This example sets the storage value deadband to 5 percent of the engineering unit range for the analog tag "BoilerTag:"

```
xp_SetStorageValueDeadband 5, 'BoilerTag'
```

xp_SetStoreForwardEvent

Starts store-and-forward data processing on the Wonderware Historian.

Syntax

```
xp_SetStoreForwardEvent
```

Remarks

This extended stored procedure is useful if you have manually copied store-and-forward blocks from the IDAS computer to the historian computer.

Extended Stored Procedure Arguments

Most of the extended stored procedures for the Wonderware Historian use one or more of the following arguments:

StartTime, EndTime

The StartTime string value represents the starting timestamp for the data to query. The EndTime string value represents the ending timestamp for the data to query. The date/time value can be any valid SQL Server date/time string.

The notion of specifying a time zone is not supported. All date/time strings passed as parameters to an extended stored procedure are considered as local server time.

For start and end times, the GetDate() and DateAdd(...) functions are supported, as well as literal dates. For more information, see "Literal Date Expressions" on page 253, "GetDate() Expressions" on page 254, and "DateAdd(...) Expressions" on page 254.

The extended stored procedures round timestamps up or down to the next supported millisecond value: 0, 3, or 7. The standard four-part query and open query do not round timestamps, so if you retrieve data with the extended stored procedure and the four-query, you can have different timestamps for the same data value.

Resolution

The sampling rate, in milliseconds, for retrieving the data in cyclic mode. The system returns values stored over the requested time period at the interval specified by the resolution. For example, if you specify a 5000 ms resolution, the system queries for all data during the time period and then only returns those values that occur at each 5000 ms interval, starting with the start date and ending with the end date.

MaxRowCount

The maximum number of rows to be returned for a specified time period.

ValueDeadBand

The percentage of the difference between the minimum and maximum engineering units for the tag. Any data values that change less than the specified deadband are not stored. The value deadband applies to delta storage only. A value of 0 indicates that a value deadband will not be applied.

TimeDeadband
The minimum time, in milliseconds, between stored values for a single tag. Any value changes that occur within the time deadband are not stored. The time deadband applies to delta storage only. A time deadband of 0 indicates that the system will store the value of the tag each time it changes.

Description
The description of the history data that will be copied.

TagN
Tag1, Tag2… TagN are tagname values. Each tagname must be comma delimited and can optionally be surrounded with quotes. For example:

```
exec xp_AnalogHistory "DateAdd(HOUR, -1,
GetDate())", "GetDate()", 1000, SysTimeSec,
SysTimeMin

exec xp_DiscreteHistory "DateAdd(HOUR, -1,
GetDate())", "GetDate()", 1000, "SysPulse"
```

If a tagname is not of the same type as expected for the named stored procedure then it is ignored. For example, if you pass an analog tagname as a parameter to xp_DiscreteHistory, it will be ignored.

For information regarding valid tagnames, see "Naming Conventions for Tags" in Chapter 2, "System-Level Functionality," in your *Wonderware Historian Concepts Guide*.

Literal Date Expressions

Date expressions can be any valid SQL Server date expression. Here are some examples

```
"4/2/2001 13:00:00:00"

"4/2/2001 12:00 PM"

"2001-4-2 1:00 AM"
```

Years expressed as two digits are interpreted as years in the 1900s. The SQL Server configuration option that supports a two-digit year cutoff is not used.

GetDate() Expressions

Date/time values can have a string value expression containing the string "GetDate". This is not the same as the SQL Server GetDate() function, although the effect is the same. Example expressions are:

```
GetDate

"GetDate"

"GetDate()"
```

For example:

```
exec xp_AnalogHistory GetDate, "GetDate()", 1000,
  'SysTimeSec'
```

DateAdd(...) Expressions

Date/time values can have a string value expression containing the string "DateAdd(...)". This is not the same as the SQL Server DateAdd() function, although the effect is very similar.

Syntax

```
"DATEADD (datepart, number, date)"
```

Parameters

DatePart

Specifies on which part of the date to return a new value. The following table lists the date parts and abbreviations recognized by the extended stored procedures for the Wonderware Historian.

MILLISECOND	MS
SECOND	SS
MINUTE	MI
HOUR	HH
DAY	DD
WEEKDAY	DW
WEEK	WK
DAYOFYEAR	DY
MONTH	MM
QUARTER	QQ
YEAR	YY

Number
The value used to increment datepart. If you specify a value that is not an integer, the fractional part of the value is discarded. For example, if you specify day for datepart and 1.75 for number, date is incremented by 1.

Date
Either a literal date value without quotes (see "Literal Date Expressions" on page 253) or a GetDate() expression also without quotes (see "GetDate() Expressions" on page 254).

Example Expressions

```
"DateAdd(HOUR, -1, GetDate())"
```

```
"DateAdd(MINUTE, -30, 4/2/2001 13:00:00:00)"
```

Extended Stored Procedure Date Expression Examples

```
xp_DiscreteHistory "DateAdd(HOUR, -1, GetDate())",
  "GetDate()", 1000, 'SysPulse'
```

```
xp_DiscreteHistoryDelta "DateAdd(DAY, -1, 4/2/2001)",
  "GetDate()", 100, 'SysPulse'
```

System Extended Stored Procedures

Extended stored procedures used internally by the Wonderware Historian are:

- xp_AllowCommit
- xp_TZgetdate

xp_AllowCommit

Determines whether a dynamic reconfiguration can be committed. Used by the **aaCommitChanges** stored procedure when a dynamic reconfiguration is requested.

Syntax

```
xp_AllowCommit AllowCommit
```

Arguments

AllowCommit
A single output parameter provided by the system.

Return Values

Value	Description
0	OK to commit.
1	The system is not running or has not been started. The commit request will be ignored and clean up the ConfigStatusPending table cleaned up.
2	Cannot commit; history block in progress.
4	Cannot commit; dynamic reconfiguration in progress.
128	Cannot commit; reason unknown.

xp_TZgetdate

Returns the correct date/time for the time zone specified. Used by the **faaTZgetdate** function.

Syntax

```
xp_TZgetdate wwTimeZone
```

Argument

wwTimeZone
Time zone for which to return the date/time.

Return Value

DateTime
The correct date/time for the requested time zone.

Stored Procedures for Internal Use

Stored procedures that are used internally by the system are prefixed with "aaInternal." For example, aaInternalAnalogTagExport. Do not use these stored procedures or change them in any way. Internal stored procedures may change from release to release, and no legacy support will be provided.

- aaInternalAnalogTagExport
- aaInternalAnalogTagImport
- aaInternalAreaDataClear
- aaInternalAreaDataInsert
- aaInternalAreaVersion
- aaInternalAreaVersionInsert

- aaInternalAreaXMLInsert
- aaInternalAttributeDataInsert
- aaInternalAttrObjectDataClear
- aaInternalDiscreteTagExport
- aaInternalDiscreteTagImport
- aaInternalEngineeringUnitExport
- aaInternalEngineeringUnitImport
- aaInternalEventTagExport
- aaInternalEventTagImport
- aaInternalHistoryModTrack
- aaInternalIODriverExport
- aaInternalIODriverImport
- aaInternalIOServerExport
- aaInternalIOServerImport
- aaInternalLicensedObjectCount
- aaInternalLicenseParametersUpdate
- aaInternalLocalizedTextExport
- aaInternalMDASAnalogTagInsert
- aaInternalMDASAnalogTagUpdate
- aaInternalMDASDiscreteTagInsert
- aaInternalMDASDiscreteTagUpdate
- aaInternalMDASStringTagInsert
- aaInternalMDASStringTagUpdate
- aaInternalMessageExport
- aaInternalMessageImport
- aaInternalNSAreaUpdate
- aaInternalNSObjectPopulate
- aaInternalNSObjectUpdate
- aaInternalObjectDataInsert
- aaInternalSetServerName
- aaInternalSnapshotTagExport

- aaInternalSnapshotTagImport
- aaInternalStringTagExport
- aaInternalStringTagImport
- aaInternalSummaryOperationExport
- aaInternalSummaryOperationImport
- aaInternalSummaryTagExport
- aaInternalSummaryTagImport
- aaInternalTagDetails
- aaInternalTopicExport
- aaInternalTopicImport

Creating Stored Procedures

You can create your own stored procedures for use with the Wonderware Historian. All procedure names will be stored in the Runtime database. The stored procedure text will be stored in the Microsoft SQL Server and retrieved at startup (from procedures created in an older session) as well as at creation time (from procedures created in the current session). Temporary procedures will not be supported. No arguments are allowed.

As with Microsoft SQL Server support, support for dynamic stored procedures for the historian will be such that when defining a stored procedure, you can create a stored procedure only in the current database, and the CREATE PROCEDURE statement cannot be combined with other SQL statements in a single batch.

Creating your own stored procedures is useful when you want to execute certain types of queries through a typical ODBC connection. The historian requires a specific ODBC configuration unless you create a stored procedure to execute the query.

For example, the following query creates a stored procedure that returns the timestamp and value for the tag 'ReactLevel" for the last 15 minutes.

```
CREATE PROCedure MyProc

AS

SELECT DateTime, TagName, Value
    FROM History
        WHERE TagName = 'ReactLevel'
            AND DateTime >= DATEADD(mi, -15, GETDATE())
            AND DateTime <= GETDATE()
```

Chapter 5

User-Defined Functions

A SQL Server function is a subroutine containing one or more Transact-SQL statements. Functions can be used to encapsulate code for reuse.

faaCheckLicenseViolation

Checks to see if the total number of tags in the Wonderware Historian is less than or equal to the number allowed by the current license.

Syntax

```
SELECT dbo.faaCheckLicenseViolation()
```

Return Type

Integer.

Remarks

If the total number of tags in the system is below the amount allowed, the result of this function will be 0. If not, the number of tags that exceed the allocated amount will be returned. For example, if a system has 100 tags, but the license only allows for 60, a value of 40 will be returned.

faaContainedName

Given a string in the form of "TagName [ContainedName]," returns the ContainedName.

Syntax

```
SELECT dbo.faaContainedName(DisplayName)
```

Arguments

DisplayName
The name as it appears in the model view hierarchy. The display name format is: TagName [ContainedName].

Return Type

Nvarchar(255).

Remarks

The maximum number of characters for both the display name and the returned contained name is 255.

faaGetHistorianTagNames

Returns the historian tagname, when provided an ArchestrA hierarchical attribute name starting with tagname as the input.

Syntax

```
SELECT dbo.faaGetHistorianTagNames(
  HierarchicalAttributeName)
```

Arguments

HierarchicalAttributeName
an ArchestrA hierarchical attribute name starting with tagname as the input.

faaGetHierarchicalAttributeNames

Returns the ArchestrA hierarchical name plus the attribute name, when provided a historian tagname.

Syntax

```
SELECT dbo.faaGetHierarchicalAttributeNames(
  HistorianTagname)
```

Arguments

HistorianTagname
Tagname within the historian for which you want to return the hierarchical name.

faaGetLocalizedText

Returns the strings from the LocalizedText table for the requested language. If the specified translation is not found, English strings are returned.

Syntax

```
SELECT * FROM dbo.faaGetLocalizedText(LangID)
```

Arguments

LangID
The locale ID for the language used. This ID is also used in the SQL Server syslanguages table.

Table Returned

The results are returned as a table that has the same columns as the LocalizedText table. However, the returned table will only include those rows containing strings in the specified language.

faaLicensedTagDetails

Returns the total number of tags and the number of licensed tags in the system, as well as for each tag type.

Syntax

```
SELECT * FROM dbo.faaLicensedTagDetails()
```

Table Returned

The result is returned in a table format. For example:

Tag Type	Tag Count	Licensed Tags
Analog	213	121
Discrete	68	60
String	27	26
Event	3	0
Total	311	207

Remarks

System tags and event tags are not included in the total tag count for licensing purposes.

faaLicensedTagTotal

Returns the total number of tags in the system for the specified tag type or for all tags.

Syntax

```
SELECT dbo.faaLicensedTagTotal(TagType)
```

Arguments

TagType
The type of tag. 1 = Analog; 2 = Discrete; 3 = String; 4 = Complex; 0 = All tags.

Return Type

Integer.

Remarks

System tags and event tags are not included in the total tag count for licensing purposes.

faaObjectTagName

Given a string in the form of "TagName [ContainedName]," returns the Tagname.

Syntax

```
SELECT dbo.faaObjectTagName(DisplayName)
```

Arguments

DisplayName
The name as it appears in the model view hierarchy. The display name format is: TagName [ContainedName].

Return Type

Nvarchar(255).

Remarks

The maximum number of characters for both the display name and the returned tagname is 255.

faaTagsInLicenseViolation

Returns information about tags that have been disabled due to violation of the licensed tag count. The information is returned in a table format.

Syntax

```
SELECT * FROM dbo.faaTagsInLicenseViolation()
```

Table Returned

Column	Data type	Description
TagName	nvarchar(256)	The unique name of the tag within the Wonderware Historian system.
wwTagKey	int	The unique numerical identifier of a tag within a single Wonderware Historian.
Description	nvarchar(512)	The description of the tag.
Address	nvarchar(512)	The address information for the tag, which consists of the computer name, application name, topic, and item name. For example, \\kc1\VIEW!Tagname:ReactLevel.

Remarks

If the licensed tag count has been exceeded, the historian will disable enough tags to run with the allowed licensed tag count. To determine the tags that are in violation of the license, the system first generates the total number of analog, discrete, complex, and string tags. (System and event tags are not included in the total tag count for licensing.) If the total number of tags exceeds the number allowed by the license, the most recent tag additions to the system are disabled. The most recent additions are indicated by the wwTagKey column of the Tag table; the higher the number, the more recent the addition.

faaTZgetdate

Returns the date/time, in the appropriate time zone.

Syntax

```
SELECT dbo.faaTZgetdate(TimeZone)
```

Arguments

TimeZone
The name of the time zone.

Return Type

DateTime.

Remarks

Use this function instead of the SQL **GetDate()** function to specify a time zone other than the server time zone in a query. To retrieve data in the time zone of the server, just use the SQL **GetDate()** function.

Example

```
DECLARE @starttime datetime

SET @starttime = dbo.faaTZgetdate('eastern daylight
  time')

SELECT DateTime, TagName, Value FROM History
    WHERE TagName IN ('SysTimeHour', 'SysTimeMin',
    'SysTimeSec')
       AND DateTime > DateAdd(mi, -30, @starttime)
       AND DateTime < DateAdd(mi, -5, @starttime)
       AND wwTimeZone = 'eastern daylight time'
```

For more information on using date/time functions in a query, see "Using DateTime Functions" in Chapter 8, "Query Examples," in your *Wonderware Historian Concepts Guide*.

faaUser_ID

Returns the database user ID (in the Runtime database) for the current user, if the user has an individual login. Returns the database ID of the appropriate Windows security group, if the current user is a group member and does have an individual login (that is, the current user logs in by virtue of being a member of the Windows group).

Syntax

```
SELECT dbo.faaUser_ID()
```

Return Type

Integer.

Remarks

This function is used for processing annotations and for support of private namespaces.

- If a user has their own database login, the user has a completely private namespace and private annotations.
- If the user is part of a Windows security group, and logs in only by virtue of being part of the group, the private namespace and annotations are shared with all members of that group.

This stored procedure assumes that Windows users that are logged in are only members of a single Windows group (configured in SQL Server). If a user is found in more than one group, the ID of the last group found is used. This could be a problem if you are expecting a given user to have access to a particular private group.

Also, it is possible that annotations and namespace entries are created under names that you might not expect. For example, a user is a local administrator on a computer, and the user's login has also been added to the aaUsers local group. When logging in to SQL Server, the user will be mapped to the sysadmin fixed server role, by virtue of the user's membership in the BUILTIN\Administrators group. (This assumes that the BUILTIN\Administrators login has not been modified or disabled for security reasons). If this user creates annotations or private namespace entries, these appear as if they had been created by "dbo," rather than by a member of the local aaUsers group.

Chapter 6

Backward Compatibility Entities

Some entities are included in the database for backward compatibility support only. It is recommended that you begin to discontinue the use of these entities.

Backward Compatibility Views

Backward compatibility views include:

- History Table Views
- Summary Views
- NamedSystemParameter
- SystemNameSpace
- InSQLSysObjects
- v_ErrorLog

History Table Views

All of the following views reflect the same table structure as the extension tables after which they are named.

These views:	Reference this extension table:
AnalogHistory, v_AnalogHistory	INSQL.Runtime.dbo.AnalogHistory
AnalogLive, v_AnalogLive	INSQL.Runtime.dbo.AnalogLive
DiscreteHistory, v_DiscreteHistory	INSQLD.Runtime.dbo.DiscreteHistory
DiscreteLive v_DiscreteLive	INSQLD.Runtime.dbo.DiscreteLive

These views:	Reference this extension table:
v_History	INSQL.Runtime.dbo.History
v_HistoryBlock	INSQL.Runtime.dbo.HistoryBlock
v_Live	INSQL.Runtime.dbo.Live
StringHistory, v_StringHistory	INSQL.Runtime.dbo.StringHistory
StringLive, v_StringLive	INSQL.Runtime.dbo.StringLive

To allow joins between the analog, string, and discrete tables, the analog and string views reference the OLE DB linked server "INSQL," while the discrete views reference the OLE DB linked server "INSQLD."

Note In SQL Server Management Studio, the extension tables are listed under the INSQL or INSQLD linked servers under the **Server objects** tree item.

Summary Views

The summary views allow you to query for data that was summarized by the event subsystem. Each of the views contains data for a specific source, frequency, and operation.

View	Contains One Row For Each
DynDailyAvg	Daily average value for a tag.
DynDailySum	Daily summary value for a tag.
DynHourlyAvg	Hourly average value for a tag.
DynHourlyMax	Hourly maximum value for a tag.
DynHourlyMin	Hourly minimum value for a tag.
DynHourlySum	Hourly summary value for a tag.
v_SummaryData	Returns one row for each summarization of a tag for an associated summary event tag.

Each table view contains the following columns:

Column	Data type	Description
TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system.

Column	Data type	Description
SummaryDate	datetime/datetime2 NOT NULL	The date applicable to the results of the calculation. It is either the time of the beginning or end of the calculation period, as specified by the summary operation definition. (If you are using Wonderware Historian with SQL Server 2008, the data type is datetime2. For more information, see "Millisecond Resolution Differences in SQL Server 2008" on page 135.)
Value	float(8) NULL	The value of the summary.
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.

v_SummaryData

Returns one row for each summarization of a tag (specified by the TagName column) for an associated summary event tag (specified by the EventTag column). The resolution is applied to data before the calculation is performed. The quality value returned is the highest quality value of the raw data from which the result is calculated..

Column	Data type	Description
TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system.
CalcType	varchar(3) NULL	The type of calculation to be performed: SUM, MAX, MIN, or AVG.
SummaryDate	datetime NOT NULL	The date applicable to the results of the calculation. It is either the time of the beginning or end of the calculation period, as specified by the summary operation definition.
Value	float(8) NULL	The value of the summary.
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.
Duration	real NULL	The period, in seconds, for which the calculation is performed.

Column	Data type	Description
Resolution	int NULL	The sampling rate, in milliseconds, for retrieving the data in cyclic mode. The system returns values stored over the requested time period at the interval specified by the resolution. For example, if you specify a 5000 ms resolution, the system queries for all data during the time period and then only returns those values that occur at each 5000 ms interval, starting with the start date and ending with the end date.
TimeStamp	tinyint NULL	The timestamp to use when storing the result of the calculation. The timestamp can be either the time when the calculation period starts or ends.
EventTag	nvarchar(256) NOT NULL	The name of the event tag to which the snapshot tag is related.

NamedSystemParameter

Contains one row for each system parameter. This view provides backward compatibility support for the NamedSystemParameter table, which has been replaced by the SystemParameter table.

Column	Data type	Description
Name	nvarchar(50) NOT NULL	The unique name for the system parameter.
Type	varchar(7) NULL	Used to specify the datatype for the system parameter value. Valid values are: NUMERIC, STRING.
StringValue	varchar(255) NULL	The value of the system parameter. This column only contains values of type STRING..
NumericValue	real NULL	The value of the system parameter. This column only contains values of type NUMERIC.

Column	Data type	Description
Editable	bit NULL	Used to determine if the value of the named system parameter can be changed using the InSQL Console. 1 = Editable; 0 = Not editable.
Description	nvarchar(255) NULL	The description of the system parameter.

SystemNameSpace

Contains one row for each item in a single system namespace. Items in the system namespace include servers, topics, and users. The items are organized in a hierarchy. This view provides backward compatibility support for the SystemNameSpace table, which has been deleted.

Column	Data type	Description
NameKey	int NULL	The unique identifier for the object in the namespace.
Type	int NOT NULL	The value that specifies the type of namespace. 1 to 6 = Tag; 1 to 2 million = System; 2+ million = Groups. Within the system range, the following values designate ArchestrA object types: 1999023 = Galaxy; 1999001 = WinPlatform object; 1999003 = AppEngine object; 1999013 = Area object; 1999011 = DDESuiteLinkClient, OPCClient, and InTouchProxy objects; 1999024 = RedundantDIObject object; 1999033 = Undeployed object represented by a generic name; 1999901 = ApplicationObject; 1999902 = Traceability object.
ParentKey	int NULL	The unique identifier for a named object in this namespace.
Name	nvarchar(290) NULL	The name of this object in the hierarchy.
PKey	int NULL	The primary key reference for other tables.

InSQLSysObjects

Contains one row for each object in the database for which changes can be tracked. This view provides backward compatibility support for the InSQLSysObjects table, which has been renamed to HistorianSysObjects.

Column	Data Type	Description
id	int NOT NULL	The unique identifier for the object.
Type	char(2) NULL	The type of object. C = CHECK constraint; D = Default or DEFAULT constraint; F = FOREIGN KEY constraint; K = PRIMARY KEY or UNIQUE constraint; L = Log; P = Stored procedure; R = Rule; RF = Stored procedure for replication; S = System table; TR = Trigger; U = User table; V = View; X = Extended stored procedure. Currently, only changes for the user tables (object type U) are tracked.
Name	varchar(50) NULL	The name of the modified object.

v_ErrorLog

Contains one row for each system message (error message), if this functionality was enabled. By default, this table is not used. For more information, see Chapter 10, "Monitoring the System," in your *Wonderware Historian Administration Guide*.

Column	Data type	Description
DateTime	datetime NOT NULL	The date that the message was written to the system log, in the local time of the Wonderware Historian.
Type	nvarchar(10) NULL	The type of system message.
LocalizedText	nvarchar(256) NULL	The content of the message.
Parameter	nvarchar(256) NULL	Optional details pertaining to the message text. For example, for the message "Disk space remaining on circular path" the parameter would contain the number of MB.

Column	Data type	Description
TotalCount	int NULL	Used to prevent "flooding" conditions in the log file. If a particular message is generated numerous times during a relatively short period of time, the message is written to the log file only once, and the total number of times that it occurred appears in this column.
ModuleID	int NULL	A unique number assigned to the Wonderware Historian subsystem that generated the message.
Host	nvarchar(32) NULL	The computer on which the Wonderware Historian subsystem runs.
FileName	nvarchar(64) NULL	Used to indicate the program file that contains the line of code that an error message comes from. Used for debugging.
Line	int NULL	Used to indicate the line of code that an error message comes from. Used for debugging.

Backward Compatibility Tables

The backward compatibility tables include:

- AnalogHistory (INSQL.Runtime.dbo.AnalogHistory)
- AnalogLive (INSQL.Runtime.dbo.AnalogLive)
- AnalogWideHistory
- ComplexHistory
- DiscreteHistory (INSQL.Runtime.dbo.DiscreteHistory)
- DiscreteLive (INSQL.Runtime.dbo.DiscreteLive)
- DiscreteWideHistory
- GroupTagList
- ManualAnalogHistory
- ManualDiscreteHistory
- ManualStringHistory
- NameSpaceIcons
- StringHistory (INSQL.Runtime.dbo.StringHistory)

- StringLive (INSQL.Runtime.dbo.StringLive)
- StringWideHistory
- WideTableDictionary

AnalogHistory (INSQL.Runtime.dbo.AnalogHistory)

This table has been superceded by the History table. For descriptions of columns in this table, see "History (INSQL.Runtime.dbo.History)" on page 52.

Column	Data type
DateTime	datetime NOT NULL
TagName	nvarchar(256) NOT NULL
Value	float(8) NULL
Quality	tinyint NOT NULL
QualityDetail	int NULL
OPCQuality	int NULL
wwTagKey	int NOT NULL
wwRowCount	int NULL
wwResolution	int NULL
wwEdgeDetection	nvarchar(16) NULL
wwRetrievalMode	nvarchar(20) NULL
wwTimeDeadband	int NULL
wwValueDeadband	real NULL
wwTimeZone	nvarchar(50) NULL
wwVersion	nvarchar(30) NULL
wwCycleCount	int NULL
wwTimeStampRule	nvarchar(20) NULL
wwInterpolationType	nvarchar(20) NULL
wwQualityRule	nvarchar(20) NULL
wwParameters	nvarchar(128) NULL

AnalogLive (INSQL.Runtime.dbo.AnalogLive)

This table has been superceded by the Live table. For descriptions of columns in this table, see "Live (INSQL.Runtime.dbo.Live)" on page 72.

Column	Data type
DateTime	datetime NOT NULL
TagName	nvarchar(256) NOT NULL
Value	float(8) NULL
Quality	tinyint NOT NULL
QualityDetail	int NULL
OPCQuality	int NULL
wwTagKey	int NOT NULL
wwRetrievalMode	nvarchar(20) NULL
wwTimeDeadband	int NULL
wwValueDeadband	real NULL
wwTimeZone	nvarchar(50) NULL
wwParameters	nvarchar(128) NULL

AnalogWideHistory

This table (INSQL.Runtime.dbo.AnalogWideHistory) has been superceded by the WideHistory table. AnalogWideHistory is the wide version of AnalogHistory. In a query, this table must be referenced using an OPENQUERY statement. For descriptions of columns in this table, see "WideHistory (INSQL.Runtime.dbo.WideHistory)" on page 119.

Column	Data type
DateTime	datetime NOT NULL
TagA1	float(8) NULL
TagA2	float(8) NULL
ManyOtherTags	float(8) NULL
wwRowCount	int NULL
wwResolution	int NULL
wwEdgeDetection	nvarchar(16) NULL

Column	Data type
wwRetrievalMode	nvarchar(20) NULL (default wwRetrievalMode is CYCLIC)
wwTimeDeadband	int NULL
wwValueDeadband	real NULL
wwTimeZone	nvarchar(50) NULL
wwVersion	nvarchar(30) NULL
wwCycleCount	int NULL
wwTimeStampRule	nvarchar(20) NULL
wwInterpolationType	nvarchar(20) NULL
wwQualityRule	nvarchar(20) NULL
wwParameters	nvarchar(128) NULL

ComplexHistory

Contains one row for each stored complex tag over time. Complex tags include BLOBs such as arrays, .AVI files, bitmaps, and so on. The ComplexHistory table is not automatically populated by the system, but rather by a client application. The ComplexHistory table is a normal SQL Server table.

Note Complex tags are currently not supported. BLOBs are stored as strings.

Column	Data type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system.
DateTime	datetime NOT NULL	The timestamp reflecting when the complex history data was acquired.
Content	image NULL	The content to be stored as a complex value. Content can be arrays, .AVI files, bitmaps, and so on.
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.
QualityDetail	int NULL	An internal representation of data quality.

ComplexTag

Contains one row for each defined complex tag. Configuration information particular to complex tags is stored in this table, while general information for all tag types is stored in the Tag table. A complex tag is any type of tag data other than analogs, discretes, or strings. Examples are arrays and video.

Note Complex tags are currently not supported. BLOBs are stored as strings.

Column	Data type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system.
InitialValue	image NULL	The initial value as imported from an external source (for example, from InTouch).
Provider	nvarchar(128) NULL	The name of the function that performs a specific operation on the data (content) before the data is stored in the database. For example, a compression algorithm. Provider functions are stored in a Wonderware Historian .DLL.
Retriever	nvarchar(128) NULL	The name of the function that performs a specific operation on the data (content) before the data can be retrieved from the database. For example, an algorithm to uncompress data. Retriever functions are stored in a Wonderware Historian .DLL.
Type	int NOT NULL	The type of complex tag. 0 = COM Object; 1 = Data operated upon by functions; 2 = Data treated as a file for use by an executable.

DiscreteHistory (INSQL.Runtime.dbo.DiscreteHistory)

This table has been superceded by the History table. For descriptions of columns in this table, see "History (INSQL.Runtime.dbo.History)" on page 52.

Column	Data type
DateTime	datetime NOT NULL
TagName	nvarchar(256) NOT NULL
Value	float(8) NULL
Quality	tinyint NOT NULL
QualityDetail	int NULL
OPCQuality	int NULL
wwTagKey	int NOT NULL
wwRowCount	int NULL
wwResolution	int NULL
wwEdgeDetection	nvarchar(16) NULL
wwRetrievalMode	nvarchar(20) NULL
wwTimeDeadband	int NULL
wwTimeZone	nvarchar(50) NULL
wwVersion	nvarchar(30) NULL
wwCycleCount	int NULL
wwTimeStampRule	nvarchar(20) NULL
wwParameters	nvarchar(128) NULL
wwQualityRule	nvarchar(20) NULL

DiscreteLive (INSQL.Runtime.dbo.DiscreteLive)

This table has been superceded by the Live table. For descriptions of columns in this table, see "Live (INSQL.Runtime.dbo.Live)" on page 72.

Column	Data type
DateTime	datetime NOT NULL
TagName	nvarchar(256) NOT NULL

Column	Data type
Value	float(8) NULL
Quality	tinyint NOT NULL
QualityDetail	int NULL
OPCQuality	int NULL
wwTagKey	int NOT NULL
wwRetrievalMode	nvarchar(20) NULL The default mode is DELTA. No other retrieval mode is allowed.
wwTimeDeadband	int NULL
wwTimeZone	nvarchar(50) NULL
wwParameters	nvarchar(128) NULL

DiscreteWideHistory

This table (INSQL.Runtime.dbo.DiscreteWideHistory) has been superceded by the WideHistory_OLEDB table. DiscreteWideHistory is the wide version of DiscreteHistory, where only discrete tags are considered. It is the same as WideHistory applied to discrete tags. In a query, this table must be referenced using an OPENQUERY statement. For descriptions of columns in this table, see "WideHistory (INSQL.Runtime.dbo.WideHistory)" on page 119.

Column	Data type
DateTime	datetime NOT NULL
TagD1	tinyint NULL
TagD2	tinyint NULL
ManyOtherTags	tinyint NULL
wwRowCount	int NULL
wwResolution	int NULL
wwEdgeDetection	nvarchar(16) NULL
wwRetrievalMode	nvarchar(20) NULL The default is DELTA.
wwTimeDeadband	int NULL

Column	Data type
wwTimeZone	nvarchar(50) NULL
wwVersion	nvarchar(30) NULL
wwCycleCount	int NULL
wwTimeStampRule	nvarchar(20) NULL
wwParameters	nvarchar(128) NULL

GroupTagList

Contains one row for each defined group of tags.

Column	Data type	Description
(FK) GroupID	int NOT NULL	The identifier for a group of tags.
(FK) wwDomainTagKey	int NOT NULL	The unique numerical identifier for a tag in a specific domain.
Triggerval	float(8) NULL	A value that can be read by an application as a trigger value.

ManualAnalogHistory

This table can be used by custom client applications to store values for analog tags. By default, this table is empty. If written to by a client application, this table will contain one row for each defined analog tag per sample period. ManualAnalogHistory is a normal SQL Server table and does not support any of the Wonderware Historian extensions for handling data.

Column	Data type	Description
DateTime	datetime NOT NULL	The timestamp reflecting when the data was acquired or stored.
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system.
Value	float(8) NULL	The value of the tag at the timestamp. Measured in engineering units.
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.

Column	Data type	Description
QualityDetail	int NULL	An internal representation of data quality.
wwTagKey	int NOT NULL	The unique numerical identifier of a tag within a single Wonderware Historian.

ManualDiscreteHistory

This table can be used by custom client applications to store values for discrete tags. By default, this table is empty. If written to by a client application, this table will contain one row for each defined discrete tag per sample period. ManualDiscreteHistory is a normal SQL Server table and does not support any of the Wonderware Historian extensions for handling data.

Column	Data type	Description
DateTime	datetime NOT NULL	The timestamp reflecting when the data was acquired or stored.
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system.
Value	tinyint NULL	The value of the discrete tag at timestamp. 0 = FALSE; 1 = TRUE; NULL = No data.
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.
QualityDetail	int NULL	An internal representation of data quality.
wwTagKey	int NOT NULL	The unique numerical identifier of a tag within a single Wonderware Historian.

ManualStringHistory

This table can be used by custom client applications to store values for string tags. By default, this table is empty. If written to by a client application, this table will contain one row for each defined string tag per sample period.

ManualStringHistory is a normal SQL Server table and does not support any of the Wonderware Historian extensions for handling data.

Column	Data type	Description
DateTime	datetime NOT NULL	The timestamp reflecting when the data was acquired or stored.
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system.
Value	nvarchar(512) NULL	The value of the string tag at the timestamp.
Quality	tinyint NOT NULL	The basic data quality indicator associated with the data value.
QualityDetail	int NULL	An internal representation of data quality.
wwTagKey	int NOT NULL	The unique numerical identifier of a tag within a single Wonderware Historian.

NameSpaceIcons

Contains one row for each defined namespace icon. Namespace icons can be shown in an application browser for each level of the namespace (system, public, and private).

Column	Data type	Description
Type	int NOT NULL	The value that specifies the type of namespace. 1 to 6 = Tag; 1 to 2 million = System; 2+ million = Groups. Within the system range, the following values designate ArchestrA object types: 1999023 = Galaxy; 1999001 = WinPlatform object; 1999003 = AppEngine object; 1999013 = Area object; 1999011 = DDESuiteLinkClient, OPCClient, and InTouchProxy objects; 1999024 = RedundantDIObject object; 1999033 = Undeployed object represented by a generic name; 1999901 = ApplicationObject; 1999902 = Traceability object.

Column	Data type	Description
Icon	image NULL	The icon for the namespace.
Name	nvarchar(30) NOT NULL	The name of the icon.
Description	nvarchar(50) NULL	The description of the icon.

StringHistory (INSQL.Runtime.dbo.StringHistory)

This table has been superceded by the History table. For descriptions of columns in this table, see "History (INSQL.Runtime.dbo.History)" on page 52.

Column	Data type
DateTime	datetime NOT NULL
TagName	nvarchar(256) NOT NULL
Value	nvarchar(512) NULL
Quality	tinyint NOT NULL
QualityDetail	int NULL
OPCQuality	int NULL
wwTagKey	int NOT NULL
wwRowCount	int NULL
wwResolution	int NULL
wwEdgeDetection	nvarchar(16) NULL
wwRetrievalMode	nvarchar(20) NULL
wwTimeDeadband	int NULL
wwTimeZone	nvarchar(50) NULL
wwVersion	nvarchar(30) NULL
wwCycleCount	int NULL
wwTimeStampRule	nvarchar(20) NULL
wwParameters	nvarchar(128) NULL
wwQualityRule	nvarchar(20) NULL

StringLive (INSQL.Runtime.dbo.StringLive)

This table has been superceded by the Live table. For descriptions of columns in this table, see "Live (INSQL.Runtime.dbo.Live)" on page 72.

Column	Data type
DateTime	datetime NOT NULL
TagName	nvarchar(256) NOT NULL
Value	nvarchar(512) NULL
Quality	tinyint NOT NULL
QualityDetail	int NULL
OPCQuality	int NULL
wwTagKey	int NOT NULL
wwRetrievalMode	nvarchar(20) NULL The default mode is DELTA. No other retrieval mode is allowed.
wwTimeDeadband	int NULL
wwTimeZone	nvarchar(50) NULL
wwParameters	nvarchar(128) NULL

StringWideHistory

This table (INSQL.Runtime.dbo.StringWideHistory) was implemented for completeness. However, you should use the WideHistory table instead of this table. In a query, this table must be referenced using an OPENQUERY statement. For descriptions of columns in this table, see "WideHistory (INSQL.Runtime.dbo.WideHistory)" on page 119.

Column	Data type
DateTime	datetime NOT NULL
TagS1	nvarchar(512) NULL
TagS2	nvarchar(512) NULL
ManyOtherTags	nvarchar(512) NULL
wwRowCount	int NULL
wwResolution	int NULL
wwEdgeDetection	nvarchar(16) NULL

Column	Data type
wwRetrievalMode	nvarchar(20) NULL (default wwRetrievalMode is DELTA)
wwTimeDeadband	int NULL
wwTimeZone	nvarchar(50) NULL
wwVersion	nvarchar(30) NULL
wwCycleCount	int NULL
wwTimeStampRule	nvarchar(20) NULL
wwParameters	nvarchar(128) NULL

TagGroup

Contains one row for each defined tag group. A tag group is a simple, non-hierarchical grouping of tags that can be used by the system.

Column	Data type	Description
GroupID	int NOT NULL	The identifier for a group of tags.
Description	nvarchar(50) NULL	The description for the group of tags.
CreatedDate	datetime NULL	The creation date for the tag grouping.
CreatedBy	nchar(18) NULL	The name of the user or application that created the group of tags.
Type	int NULL	The type of tag group. 1 to 100 = System use. 100 = Users and third-party client applications.

WideTableDictionary

Contains one row of values for up to 249 tags. These 249 tags appear as columns that will be visible in the data dictionary for each user. Mainly used by ad-hoc query tools. Does not affect the ability to access the values stored for a tag.

Column	Data type	Description
(FK) UserKey	int NOT NULL	The unique numerical identifier for a database user as defined in the UserDetail table. UserKey is a foreign key from the UserDetail table.

Column	Data type	Description
(FK) TagName	nvarchar(256) NOT NULL	The unique name of the tag within the Wonderware Historian system. TagName is a foreign key from the Tag table.

Renamed Tables

The following table has been renamed. A view named InSQLSysObjects has been created for backward compatibility.

Old Name	New Name
InSQLSysObjects	HistorianSysObjects

Backward Compatibility Stored Procedures

Stored procedures that have been retained for backward compatibility are:

- aaAnalogDetail
- aaDiscreteDetail
- aaStringDetail
- ww_CheckClientVersion
- ww_CheckWhichDb
- ww_dbCheck
- ww_LoadInSQLProcedureBody
- ww_MDASAnalogTagInsert
- ww_MDASAnalogTagUpdate
- ww_MDASDiscreteTagInsert
- ww_MDASDiscreteTagUpdate
- ww_MDASStringTagInsert
- ww_MDASStringTagUpdate

The following logins are provided for backward compatibility only:

Login Name	Password	Description
wwUser	wwUser	Same as aaUser.
wwPower	wwPower	Same as aaPower.
wwAdmin	wwAdmin	Same as aaAdmin.
wwdbo	pwddbo	Same as aadbo.

aaAnalogDetail

Returns information about one or more specified analog tags, including the name of the tag, a description, the acquisition rate, the engineering unit, and the minimum and maximum values in engineering units.

Syntax

```
aaAnalogDetail TagList
```

Arguments

TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

Permission

Execute permission defaults to the **public** group.

aaDiscreteDetail

Returns information about one or more specified discrete tags, including the name of the tag, a description, the message for the TRUE (1) state of the tag, and the message for the FALSE (0) state of the tag.

Syntax

```
aaDescreteDetail TagList
```

Arguments

TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

Permission

Execute permission defaults to the **public** group.

aaStringDetail

Returns a description for one or more specified tags.

Syntax

```
aaStringDetail TagList
```

Arguments

TagList
A list of tags delimited by a comma (,). This value is of data type nvarchar(4000), with no default.

Permission

Execute permission defaults to the **public** group.

ww_CheckClientVersion

Checks which version of the client application is running.

Syntax

```
ww_CheckClientVersion AppName, AppVersion
```

Arguments

AppName
The name of the application. This value is of data type nvarchar(255), with a default of an empty string.

AppVersion
The version number of the application. This value is of data type nvarchar(255), with a default of an empty string.

Remarks

This stored procedure is used by the Wonderware Historian to ensure that a version of a Wonderware client application will run against the database. A client application will not be allowed to run against a database version that does not support that client.

Important This stored procedure is for Wonderware use only. Do not attempt to use this stored procedure for any third-party client application.

Permission

Execute permission defaults to the **public** group.

ww_CheckWhichDb

Used to determine if querying the correct database.

Syntax

```
ww_CheckWhichDb dbType
```

Arguments

dbType
The identifier for the database. 1 = Runtime; 2 = Holding; 3 = Development. This value is of data type int, with no default.

Permission

Execute permission defaults to the **public** group.

ww_dbCheck

Used to invalidate FactorySuite 1000 clients.

Syntax

```
ww_dbCheck dbType
```

Arguments

dbType
The identifier for the database. 1 = Runtime; 2 = Holding; 3 = Development.

This value is of data type int, with no default.

Remarks

This stored procedure is only used by client applications released prior to FactorySuite 2000.

Permission

Execute permission defaults to the **public** group.

ww_DBConfig

Returns a summary of the current database configuration, such as number of tags, number of tags per type, storage configuration, event tags, and summary configuration.

Syntax

```
ww_DBConfig
```

Permission

Execute permission defaults to the **public** group.

ww_LoadInSQLProcedureBody

Used internally to track which stored procedures reference the extension tables.

Syntax

```
ww_LoadSQLProcedureBody ObjName
```

Arguments

ObjName
The name of the stored procedure to load. This value is of data type varchar(92), with no default.

Permission

Execute permission defaults to the **public** group.

ww_MDASAnalogTagInsert

Used by the Manual Data Acquisition Service to add an analog tag.

This stored procedure calls the aaInternalMDASAnalogTagInsert stored procedure, which is for internal use only.

ww_MDASAnalogTagUpdate

Used by the Manual Data Acquisition Service to update an analog tag.

This stored procedure calls the aaInternalMDASAnalogTagUpdate stored procedure, which is for internal use only.

ww_MDASDiscreteTagInsert

Used by the Manual Data Acquisition Service to add a discrete tag.

This stored procedure calls the aaInternalMDASDiscreteTagInsert stored procedure, which is for internal use only.

ww_MDASDiscreteTagUpdate

Used by the Manual Data Acquisition Service to update a discrete tag.

This stored procedure calls the aaInternalMDASDiscreteTagUpdate stored procedure, which is for internal use only.

ww_MDASStringTagInsert

Used by the Manual Data Acquisition Service to add a string tag.

This stored procedure calls the aaInternalMDASStringTagInsert stored procedure, which is for internal use only.

ww_MDASStringTagUpdate

Used by the Manual Data Acquisition Service to update a string tag.

This stored procedure calls the aaInternalMDASStringTagUpdate stored procedure, which is for internal use only.

Renamed Stored Procedures

The following stored procedures have been renamed. The old stored procedures have been retained in the system for backward compatibility.

Old Name	New Name
ww_ActionStringSelect	aaAddAnalogSummaryTag
ww_AddTag	aaAddStructureTag
ww_AnalogDetail	aaAnalogDetail
ww_AnalogTagDelete	aaAnalogTagDelete
ww_AnalogTagInsert	aaAnalogTagInsert
ww_AnalogTagSelect	aaAnalogTagSelect
ww_AnalogTagUpdate	aaAnalogTagUpdate
ww_Annotation	aaAnnotationRetrieve
ww_AnnotationDelete	aaAnnotationDelete
ww_AnnotationInsert	aaAnnotationInsert
ww_AnnotationSelect	aaAnnotationSelect
ww_AnnotationUpdate	aaAnnotationUpdate
ww_CheckClientVersion	--
ww_CheckWhichDb	--
ww_CleanupAfterCommit	aaCleanupAfterCommit
ww_CommitChanges	aaCommitChanges

Old Name	New Name
ww_CommitChangesAtStartup	aaCommitChangesAtStartup
ww_ContextDelete	aaContextDelete
ww_ContextInsert	aaContextInsert
ww_ContextSelect	aaContextSelect
ww_ContextUpdate	aaContextUpdate
ww_DBChangesPending	aaDBChangesPending
ww_dbCheck	--
ww_DBConfig	aaDBConfig
ww_DeleteOlderEvents	aaDeleteOlderEvents
ww_DeleteOlderSummaries	aaDeleteOlderSummaries
ww_DeleteTag	aaDeleteTag
ww_DetectorStringSelect	aaDetectorStringSelect
ww_DiscreteDetail	aaDiscreteDetail
ww_DiscreteTagDelete	aaDiscreteTagDelete
ww_DiscreteTagInsert	aaDiscreteTagInsert
ww_DiscreteTagSelect	aaDiscreteTagSelect
ww_DiscreteTagUpdate	aaDiscreteTagUpdate
ww_EngineeringUnitDelete	aaEngineeringUnitDelete
ww_EngineeringUnitInsert	aaEngineeringUnitInsert
ww_EngineeringUnitSelect	aaEngineeringUnitSelect
ww_EngineeringUnitUpdate	aaEngineeringUnitUpdate
ww_EventDetection	aaEventDetection
ww_EventHistory	aaEventHistorySelect
ww_EventHistoryInsert	aaEventHistoryInsert
ww_EventSnapshot	aaEventSnapshotSelect
ww_EventSnapshotInsert	aaEventSnapshotInsert
ww_EventTagDelete	aaEventTagDelete
ww_EventTagDetail	aaEventTagDetail
ww_EventTagInsert	aaEventTagInsert
ww_EventTagSelect	aaEventTagSelect

Old Name	New Name
ww_EventTagSelectAll	aaEventTagSelectAll
ww_EventTagSelectDeleted	aaEventTagSelectDeleted
ww_EventTagSelectDisabled	aaEventTagSelectDisabled
ww_EventTagSelectInserted	aaEventTagSelectInserted
ww_EventTagSelectUpdated	aaEventTagSelectUpdated
ww_EventTagUpdate	aaEventTagUpdate
ww_GetDbRevision	aaGetDbRevision
ww_GetLastTagKey	aaGetLastTagKey
ww_HistoryBlockSelect	aaHistoryBlockSelect
ww_InSQLConfigNSExpand	aaHistorianConfigNSExpand
ww_InSQLNSExpand	aaHistorianNSExpand
ww_InSQLStatusSelect	aaHistorianStatusSelect
ww_InSQLStatusSet	aaHistorianStatusSet
ww_InTouchNodeTagList	aaInTouchNodeTagList
ww_IODriverDelete	aaIODriverDelete
ww_IODriverInsert	aaIODriverInsert
ww_IODriverSelect	aaIODriverSelect
ww_IODriverUpdate	aaIODriverUpdate
ww_IOServerDelete	aaIOServerDelete
ww_IOServerInsert	aaIOServerInsert
ww_IOServerSelect	aaIOServerSelect
ww_IOServerTypeDelete	aaIOServerTypeDelete
ww_IOServerTypeInsert	aaIOServerTypeInsert
ww_IOServerTypeSelect	aaIOServerTypeSelect
ww_IOServerTypeUpdate	aaIOServerTypeUpdate
ww_IOServerUpdate	aaIOServerUpdate
ww_LimitDelete	aaLimitDelete
ww_LimitInsert	aaLimitInsert
ww_LimitNameDelete	aaLimitNameDelete
ww_LimitNameInsert	aaLimitNameInsert

Old Name	New Name
ww_LimitNameSelect	aaLimitNameSelect
ww_LimitNameUpdate	aaLimitNameUpdate
ww_LimitSelect	aaLimitSelect
ww_LimitUpdate	aaLimitUpdate
ww_LoadInSQLProcedureBody	--
ww_MessageDelete	aaMessageDelete
ww_MessageInsert	aaMessageInsert
ww_MessageSelect	aaMessageSelect
ww_MessageUpdate	aaMessageUpdate
ww_ModLogStatus	aaModLogStatus
ww_PrivateNSAddGroup	aaPrivateNSAddGroup
ww_PrivateNSAddLeaf	aaPrivateNSAddLeaf
ww_PrivateNSDeleteGroup	aaPrivateNSDeleteGroup
ww_PrivateNSDeleteLeaf	aaPrivateNSDeleteLeaf
ww_PrivateNSExpand	aaPrivateNSExpand
ww_PrivateNSSelect	aaPrivateNSSelect
ww_PrivateNSUpdateGroup	aaPrivateNSUpdateGroup
ww_PublicNSAddGroup	aaPublicNSAddGroup
ww_PublicNSAddLeaf	aaPublicNSAddLeaf
ww_PublicNSDeleteGroup	aaPublicNSDeleteGroup
ww_PublicNSDeleteLeaf	aaPublicNSDeleteLeaf
ww_PublicNSExpand	aaPublicNSExpand
ww_PublicNSSelect	aaPublicNSSelect
ww_PublicNSUpdateGroup	aaPublicNSUpdateGroup
ww_RedirectToInTouch	aaRedirectToInTouch
ww_SetAISamples	aaSetAISamples
ww_SetCalculatedAISamples	aaSetCalculatedAISamples
ww_SetStorageRule	aaSetStorageRule
ww_SetTagStorage	aaSetTagStorage
ww_SnapshotDetailSelect	aaSnapshotDetailSelect

Old Name	New Name
ww_SnapshotDetailUpdate	aaSnapshotDetailUpdate
ww_SnapToSummary	aaSnapToSummary
ww_SpaceManager	aaSpaceManager
ww_StorageLocationSelect	aaStorageLocationSelect
ww_StorageLocationUpdate	aaStorageLocationUpdate
ww_StringDetail	aaStringDetail
ww_StringTagDelete	aaStringTagDelete
ww_StringTagInsert	aaStringTagInsert
ww_StringTagSelect	aaStringTagSelect
ww_StringTagUpdate	aaStringTagUpdate
ww_SummaryActionInsert	aaSummaryActionInsert
ww_SummaryDetail	aaSummaryDetail
ww_SummaryOperationDelete	aaSummaryOperationDelete
ww_SummaryOperationInsert	aaSummaryOperationInsert
ww_SummaryOperationSelect	aaSummaryOperationSelect
ww_SummaryOperationUpdate	aaSummaryOperationUpdate
ww_SummaryTagListDelete	aaSummaryTagListDelete
ww_SummaryTagListInsert	aaSummaryTagListInsert
ww_SummaryTagListSelect	aaSummaryTagListSelect
ww_SummaryTagListUpdate	aaSummaryTagListUpdate
ww_SystemConfigNSExpand	aaSystemConfigNSExpand
ww_SystemNSExpand	aaSystemNSExpand
ww_SystemNSExpand2	aaSystemNSExpand2
ww_SystemParameterSelect	aaSystemParameterSelect
ww_SystemParameterUpdate	aaSystemParameterUpdate
ww_TagConfig	aaTagConfig
ww_TagConfigModified	aaTagConfigModified
ww_TagConfigSelect	aaTagConfigSelect
ww_TagInfo	aaTagInfo
ww_TagType	aaTagType

Old Name	New Name
ww_CommitChangesAtStartup	aaCommitChangesAtStartup
ww_ContextDelete	aaContextDelete
ww_ContextInsert	aaContextInsert
ww_ContextSelect	aaContextSelect
ww_ContextUpdate	aaContextUpdate
ww_DBChangesPending	aaDBChangesPending
ww_dbCheck	--
ww_DBConfig	aaDBConfig
ww_DeleteOlderEvents	aaDeleteOlderEvents
ww_DeleteOlderSummaries	aaDeleteOlderSummaries
ww_DeleteTag	aaDeleteTag
ww_DetectorStringSelect	aaDetectorStringSelect
ww_DiscreteDetail	aaDiscreteDetail
ww_DiscreteTagDelete	aaDiscreteTagDelete
ww_DiscreteTagInsert	aaDiscreteTagInsert
ww_DiscreteTagSelect	aaDiscreteTagSelect
ww_DiscreteTagUpdate	aaDiscreteTagUpdate
ww_EngineeringUnitDelete	aaEngineeringUnitDelete
ww_EngineeringUnitInsert	aaEngineeringUnitInsert
ww_EngineeringUnitSelect	aaEngineeringUnitSelect
ww_EngineeringUnitUpdate	aaEngineeringUnitUpdate
ww_EventDetection	aaEventDetection
ww_EventHistory	aaEventHistorySelect
ww_EventHistoryInsert	aaEventHistoryInsert
ww_EventSnapshot	aaEventSnapshotSelect
ww_EventSnapshotInsert	aaEventSnapshotInsert
ww_EventTagDelete	aaEventTagDelete
ww_EventTagDetail	aaEventTagDetail
ww_EventTagInsert	aaEventTagInsert
ww_EventTagSelect	aaEventTagSelect

Old Name	New Name
ww_EventTagSelectAll	aaEventTagSelectAll
ww_EventTagSelectDeleted	aaEventTagSelectDeleted
ww_EventTagSelectDisabled	aaEventTagSelectDisabled
ww_EventTagSelectInserted	aaEventTagSelectInserted
ww_EventTagSelectUpdated	aaEventTagSelectUpdated
ww_EventTagUpdate	aaEventTagUpdate
ww_GetDbRevision	aaGetDbRevision
ww_GetLastTagKey	aaGetLastTagKey
ww_HistoryBlockSelect	aaHistoryBlockSelect
ww_InSQLConfigNSExpand	aaHistorianConfigNSExpand
ww_InSQLNSExpand	aaHistorianNSExpand
ww_InSQLStatusSelect	aaHistorianStatusSelect
ww_InSQLStatusSet	aaHistorianStatusSet
ww_InTouchNodeTagList	aaInTouchNodeTagList
ww_IODriverDelete	aaIODriverDelete
ww_IODriverInsert	aaIODriverInsert
ww_IODriverSelect	aaIODriverSelect
ww_IODriverUpdate	aaIODriverUpdate
ww_IOServerDelete	aaIOServerDelete
ww_IOServerInsert	aaIOServerInsert
ww_IOServerSelect	aaIOServerSelect
ww_IOServerTypeDelete	aaIOServerTypeDelete
ww_IOServerTypeInsert	aaIOServerTypeInsert
ww_IOServerTypeSelect	aaIOServerTypeSelect
ww_IOServerTypeUpdate	aaIOServerTypeUpdate
ww_IOServerUpdate	aaIOServerUpdate
ww_LimitDelete	aaLimitDelete
ww_LimitInsert	aaLimitInsert
ww_LimitNameDelete	aaLimitNameDelete
ww_LimitNameInsert	aaLimitNameInsert

Old Name	New Name
ww_LimitNameSelect	aaLimitNameSelect
ww_LimitNameUpdate	aaLimitNameUpdate
ww_LimitSelect	aaLimitSelect
ww_LimitUpdate	aaLimitUpdate
ww_LoadInSQLProcedureBody	--
ww_MessageDelete	aaMessageDelete
ww_MessageInsert	aaMessageInsert
ww_MessageSelect	aaMessageSelect
ww_MessageUpdate	aaMessageUpdate
ww_ModLogStatus	aaModLogStatus
ww_PrivateNSAddGroup	aaPrivateNSAddGroup
ww_PrivateNSAddLeaf	aaPrivateNSAddLeaf
ww_PrivateNSDeleteGroup	aaPrivateNSDeleteGroup
ww_PrivateNSDeleteLeaf	aaPrivateNSDeleteLeaf
ww_PrivateNSExpand	aaPrivateNSExpand
ww_PrivateNSSelect	aaPrivateNSSelect
ww_PrivateNSUpdateGroup	aaPrivateNSUpdateGroup
ww_PublicNSAddGroup	aaPublicNSAddGroup
ww_PublicNSAddLeaf	aaPublicNSAddLeaf
ww_PublicNSDeleteGroup	aaPublicNSDeleteGroup
ww_PublicNSDeleteLeaf	aaPublicNSDeleteLeaf
ww_PublicNSExpand	aaPublicNSExpand
ww_PublicNSSelect	aaPublicNSSelect
ww_PublicNSUpdateGroup	aaPublicNSUpdateGroup
ww_RedirectToInTouch	aaRedirectToInTouch
ww_SetAISamples	aaSetAISamples
ww_SetCalculatedAISamples	aaSetCalculatedAISamples
ww_SetStorageRule	aaSetStorageRule
ww_SetTagStorage	aaSetTagStorage
ww_SnapshotDetailSelect	aaSnapshotDetailSelect

Old Name	New Name
ww_SnapshotDetailUpdate	aaSnapshotDetailUpdate
ww_SnapToSummary	aaSnapToSummary
ww_SpaceManager	aaSpaceManager
ww_StorageLocationSelect	aaStorageLocationSelect
ww_StorageLocationUpdate	aaStorageLocationUpdate
ww_StringDetail	aaStringDetail
ww_StringTagDelete	aaStringTagDelete
ww_StringTagInsert	aaStringTagInsert
ww_StringTagSelect	aaStringTagSelect
ww_StringTagUpdate	aaStringTagUpdate
ww_SummaryActionInsert	aaSummaryActionInsert
ww_SummaryDetail	aaSummaryDetail
ww_SummaryOperationDelete	aaSummaryOperationDelete
ww_SummaryOperationInsert	aaSummaryOperationInsert
ww_SummaryOperationSelect	aaSummaryOperationSelect
ww_SummaryOperationUpdate	aaSummaryOperationUpdate
ww_SummaryTagListDelete	aaSummaryTagListDelete
ww_SummaryTagListInsert	aaSummaryTagListInsert
ww_SummaryTagListSelect	aaSummaryTagListSelect
ww_SummaryTagListUpdate	aaSummaryTagListUpdate
ww_SystemConfigNSExpand	aaSystemConfigNSExpand
ww_SystemNSExpand	aaSystemNSExpand
ww_SystemNSExpand2	aaSystemNSExpand2
ww_SystemParameterSelect	aaSystemParameterSelect
ww_SystemParameterUpdate	aaSystemParameterUpdate
ww_TagConfig	aaTagConfig
ww_TagConfigModified	aaTagConfigModified
ww_TagConfigSelect	aaTagConfigSelect
ww_TagInfo	aaTagInfo
ww_TagType	aaTagType

Old Name	New Name
ww_TimeDetectorDetailInsert	aaTimeDetectorDetailInsert
ww_TimeDetectorDetailSelect	aaTimeDetectorDetailSelect
ww_TimeDetectorDetailUpdate	aaTimeDetectorDetailUpdate
ww_TopicDelete	aaTopicDelete
ww_TopicInsert	aaTopicInsert
ww_TopicSelect	aaTopicSelect
ww_TopicUpdate	aaTopicUpdate
ww_UpdateCalculatedAISamples	aaUpdateCalculatedAISamples
ww_UserAccessLevelSelect	aaUserAccessLevelSelect
ww_UserDetailUpdate	aaUserDetailUpdate

Backward Compatibility Functions

The following functions have been renamed. The old functions have been retained in the system for backward compatibility.

Old Name	New Name
fww_CheckLicenseViolation	faaCheckLicenseViolation
fww_GetLocalizedText	faaGetLocalizedText
fww_InSQLgetdate	faaTZgetdate
fww_LicensedTagDetails	faaLicensedTagDetails
fww_LicensedTagTotal	faaLicensedTagTotal
fww_TagsInLicenseViolation	faaTagsInLicenseViolation

Index

A

aaActionStringSelect stored procedure 138
aaAddAnalogSummaryTag stored procedure 138
aaAddReplicationGroup stored procedure 142
aaAddReplicationSchedule stored procedure 143
aaAddReplicationServer stored procedure 144
aaAddReplicationTagEntity stored procedure 145
aaAddStateSummaryTag stored procedure 146
aaAddStructureTag stored procedure 149
aaAddTag stored procedure 149, 152
aaAnalogDetail stored procedure 289
aaAnalogTagDelete stored procedure 154
aaAnalogTagInsert stored procedure 154
aaAnalogTagSelect stored procedure 159
aaAnalogTagUpdate stored procedure 159
aaAnnotationDelete stored procedure 160
aaAnnotationInsert stored procedure 160
aaAnnotationRetrieve stored procedure 161
aaAnnotationSelect stored procedure 162
aaAnnotationUpdate stored procedure 162
aaArchestrANSClear stored procedure 162
aaAreaData table 25, 28
aaAreaXML table 25, 28
aaAttributeData table 25, 29
aaAttributeDataPending table 25, 29
aaCleanUpAfterCommit stored procedure 163
aaCleanupSystemNotRunning stored procedure 163
aaCommitChanges stored procedure 164
aaCommitChangesAtStartup stored procedure 164
aaContextDelete stored procedure 164
aaContextInsert stored procedure 165
aaContextSelect stored procedure 165
aaContextUpdate stored procedure 165
aaDBChangesPending stored procedure 166
aaDBConfig stored procedure 167
aaDeleteOlderEvents stored procedure 167
aaDeleteOlderSummaries stored procedure 167
aaDeleteReplicationGroup stored procedure 168

aaDeleteReplicationSchedule stored procedure 168
aaDeleteReplicationServer stored procedure 169
aaDeleteReplicationTagEntity stored procedure 169
aaDeleteTag stored procedure 170
aaDetectorStringSelect stored procedure 170
aaDiscreteDetail stored procedure 170, 289
aaDiscreteTagDelete stored procedure 171
aaDiscreteTagInsert stored procedure 171
aaDiscreteTagSelect stored procedure 175
aaDiscreteTagUpdate stored procedure 175
aaEngineeringUnitDelete stored procedure 176
aaEngineeringUnitInsert stored procedure 176
aaEngineeringUnitSelect stored procedure 177
aaEngineeringUnitUpdate stored procedure 178
aaEventDetection stored procedure 178
aaEventHistoryInsert stored procedure 179
aaEventHistorySelect table 180
aaEventSnapshot stored procedure 181
aaEventSnapshotInsert stored procedure 181
aaEventTagDelete stored procedure 182
aaEventTagDetail stored procedure 182
aaEventTagInsert stored procedure 183
aaEventTagSelect stored procedure 186
aaEventTagSelectAll stored procedure 186
aaEventTagSelectDeleted stored procedure 186
aaEventTagSelectDisabled stored procedure 186
aaEventTagSelectInserted stored procedure 187
aaEventTagSelectUpdated stored procedure 187
aaEventTagUpdate stored procedure 187
aaGetAnalogSummaryTags stored procedure 188
aaGetDbRevision stored procedure 188
aaGetLastTagKey stored procedure 188
aaGetReplicationGroups stored procedure 189
aaGetReplicationNamingParameters stored procedure 190
aaGetReplicationSchedules stored procedure 190
aaGetReplicationServers stored procedure 191
aaGetReplicationTagEntities stored procedure 192
aaGetReplicationTags stored procedure 193
aaGetStateSummaryTags stored procedure 193
aaHistorianConfigNSExpand stored procedure 194
aaHistorianNSExpand stored procedure 194
aaHistorianStatusSelect stored procedure 194
aaHistorianStatusSet stored procedure 195
aaHistoryBlockSelect stored procedure 196
aaInTouchNodeTagList stored procedure 196
aaIODriverDelete stored procedure 197
aaIODriverInsert stored procedure 197
aaIODriverSelect stored procedure 200
aaIODriverUpdate stored procedure 200
aaIOServerDelete stored procedure 201
aaIOServerInsert stored procedure 201
aaIOServerSelect stored procedure 203
aaIOServerTypeDelete stored procedure 203
aaIOServerTypeInsert stored procedure 203
aaIOServerTypeSelect stored procedure 204
aaIOServerTypeUpdate stored procedure 204
aaIOServerUpdate stored procedure 205
aaLimitDelete stored procedure 205
aaLimitInsert stored procedure 205
aaLimitNameDelete stored procedure 206
aaLimitNameInsert stored procedure 207

aaLimitNameSelect stored procedure 207
aaLimitNameUpdate stored procedure 207
aaLimitSelect stored procedure 208
aaLimitUpdate stored procedure 208
aaMessageDelete stored procedure 208
aaMessageInsert stored procedure 209
aaMessageSelect stored procedure 209
aaMessageUpdate stored procedure 209
aaModLogStatus stored procedure 210
aaObjectData column 29
aaObjectData table 25
aaObjectDataPending table 25, 30
aaPrivateNSAddGroup stored procedure 210
aaPrivateNSAddLeaf stored procedure 211
aaPrivateNSDeleteGroup stored procedure 211
aaPrivateNSDeleteLeaf stored procedure 212
aaPrivateNSExpand stored procedure 212
aaPrivateNSSelect stored procedure 213
aaPrivateNSUpdateGroup stored procedure 213
aaPublicNSAddGroup stored procedure 213
aaPublicNSAddLeaf stored procedure 214
aaPublicNSDeleteGroup stored procedure 215
aaPublicNSDeleteLeaf stored procedure 215
aaPublicNSExpand stored procedure 216
aaPublicNSSelect stored procedure 216
aaPublicNSUpdateGroup stored procedure 216
aaRedirectToInTouch stored procedure 217
aaSetAISamples stored procedure 218
aaSetCalculatedAISamples stored procedure 218
aaSetServerTimeStamp stored procedure 219
aaSetStorageRule stored procedure 219
aaSetTagStorage stored procedure 221
aaSnapshotDetailSelect stored procedure 222
aaSnapshotDetailUpdate stored procedure 223
aaSnapToSummary stored procedure 223
aaSpaceManager stored procedure 224
aaStorageLocationSelect stored procedure 224
aaStorageLocationUpdate stored procedure 225
aaStringDetail stored procedure 290
aaStringTagDelete stored procedure 226
aaStringTagInsert stored procedure 227
aaStringTagSelect stored procedure 230
aaStringTagUpdate stored procedure 230
aaSummaryActionInsert stored procedure 231
aaSummaryDetail stored procedure 231
aaSummaryOperationDelete stored procedure 232
aaSummaryOperationInsert stored procedure 232
aaSummaryOperationSelect stored procedure 233
aaSummaryOperationUpdate stored procedure 234
aaSummaryTagListDelete stored procedure 234
aaSummaryTagListInsert stored procedure 235
aaSummaryTagListSelect stored procedure 236
aaSummaryTagListUpdate stored procedure 236
aaSystemConfigNSExpand stored procedure 236
aaSystemNSExpand stored procedure 237
aaSystemNSExpand2 stored procedure 238
aaSystemParameterSelect stored procedure 238
aaSystemParameterUpdate stored procedure 238
aaTagConfig stored procedure 239
aaTagConfigModified stored procedure 239
aaTagConfigSelect stored procedure 240
aaTagInfo stored procedure 240
aaTagName column 29, 30
aaTagType stored procedure 241

aaTimeDetectorDetailInsert stored procedure 241
aaTimeDetectorDetailSelect stored procedure 242
aaTimeDetectorDetailUpdate stored procedure 242
aaTopicDelete stored procedure 243
aaTopicInsert stored procedure 243
aaTopicSelect stored procedure 244
aaTopicUpdate stored procedure 244
aaUpdateCalculatedAISamples stored procedure 245
aaUserAccessLevelSelect stored procedure 246
aaUserDetailUpdate stored procedure 246
access level 246
AccessLevel column 119
AcqType argument 220
acquisition 110, 139, 147, 150, 152, 155, 171, 227
Acquisition argument 195
AcquisitionRate column 110
AcquisitionType argument 139, 147, 150, 152, 155, 171, 227
AcquisitionType column 110
action priority 51, 185
action scripts 50, 185
action string 138
ActionClassName column 31
actions
  See also event actions
ActionString argument 185
ActionString column 50
ActionType table 22, 31
ActionTypeKey argument 185
ActionTypeKey column 31, 49
active image 111, 112, 157, 158, 173, 174, 218, 229, 245
  setting samples for 218
address string 110, 153, 155, 172, 227
AIRetrievalMode argument 157, 173
AIRetrievalMode column 112
AITag column 113
alarm 44
AllowCommit argument 255
AltComputerName argument 199, 202
AltComputerName column 64, 69
AlwaysAdvise column 118
AlwaysModifyName column 62
analog tags
  deleting 154
  inserting 154, 292
  selecting 159
  updating 159, 292
AnalogHistory table 247, 276
AnalogHistory view 269
AnalogLive table 277
AnalogLive view 269
AnalogSnapshot table 22, 31
AnalogSummaryHistory table 18, 32
AnalogSummaryHistory view 126
AnalogSummaryTag table 24, 36
AnalogTag table 24, 37
AnalogWideHistory table 248, 277
Annotation table 24, 39, 40
AnnotationKey argument 160, 162
AnnotationKey column 39
annotations 39
  deleting 160
  inserting 160
  retrieving 161
  selecting 162
  updating 162
application name
  I/O Server 69, 70, 201, 203, 204
ApplicationName argument 201, 203, 204
ApplicationName column 62, 69, 70, 117
AppName argument 290
AppVersion argument 290
ArchestrA
  browsing tables 25
ArchiveDate column 60
ArchiveLocation column 60
area 28
AreaKey column 28
AreaName column 28
AreaXML column 28
attribute
  data 29
AttributeName column 29, 103
AttributeOrder column 103
AttributeType table 40
AttributeTypeKey column 40, 103
AttributeTypeName column 40
AttributeTypeValue column 40

AuthenticateWithAAUser column 83, 133
autonomous startup 67, 199
AutonomousStartupTimeout argument 199
AutonomousStartupTimeout column 67
AutoStart argument 202
AutoStart column 69
Average column 35

B

backward compatibility
  database entities 269
  stored procedures 288
  tables 275
Bandwidth column 84, 133
blocks
  See history blocks
BufferCount argument 199
BufferCount column 67, 84, 133
buffers
  IDAS 67, 199

C

CalcType argument 233
CalcType column 40, 107, 271
CalcType table 22, 40
CalculatedAISamples column 112
calculation 40, 107, 233, 271
Category column 28
CEVersion column 113
characters
  limit for string tags 103, 228
Checked argument 206
Checked column 71, 80
Column column 128
ColumnName column 75
Comment column 63
complex tags 278, 279
ComplexHistory table 278
ComplexTag table 279
Compression column 60
ComputerName argument 197, 202
ComputerName column 64, 69, 89, 101
ConfigStatusPending table 25, 40
ConfigStatusSnapshot table 25, 41
ConfigStor column 78, 79
configuration
  See also dynamic configuration
configuration tables
  about 25
ConnectionTimeout argument 200
ConnectionTimeout column 68
ContainedName column 28, 30
ContainedStateCount column 92
Content argument 161
Content column 39, 278
Context table 24, 41, 42
ContextKey argument 164, 165, 205
ContextKey column 41, 42, 44, 70, 80
contexts 41, 165
  deleting 164
  inserting 165
  selecting 165
  updating 165
counter 39, 158
CreatedBy argument 139, 147, 149, 155, 172, 183, 228
CreatedBy column 111, 287
CreatedDate column 287
CreateGroup column 82
CreateReplicationServerDefaultGroups stored procedure 166
CreateReplicationServerSystemTags stored procedure 166
current values 72
CurrentEditor argument 141, 148, 151, 153, 156, 173, 183, 229
CurrentEditor column 42, 88, 111, 132
CustomReplicationSchedule table 42
cycle count 55, 97, 122

D

data acquisition
  See acquisition
data blocks
  See history blocks
data dictionary 62, 63
data files
  See history blocks
data quality 32, 45, 52, 72, 102, 104, 127, 128, 130, 271, 278, 282, 283, 284
  See also quality
data retrieval
  See retrieval

data storage
See storage
database
revision number 188
database configuration
returning summary 167, 291
database modifications 40
database user 76, 119, 129, 266
date expression
extended stored procedures 253, 254
DateCreated argument 139, 147, 149, 155, 160, 172, 183, 228
DateCreated column 39, 110
DateStamp argument 224
DateTime argument 160, 179
DateTime column 39, 47, 48, 52, 72, 76, 95, 119, 128, 274, 276, 277, 278, 280, 281, 282, 283, 284, 285, 286
datetime2 data type 135
daylight savings time 55, 60, 73, 97, 116, 122
DbError column 101
DbModAcquisition column 101
DbModAll column 102
DbModServer column 101
DbModStorage column 101
DbRevision column 102
DBServer argument 195
DbStatus argument 195
DbStatus column 101
DBType argument 220
dbType argument 291
DDE access name 117
DdeSourceKey column 117
deadband 38, 54, 96, 97, 110, 118, 121, 141, 148, 151, 153, 155, 157, 172, 184, 220, 228, 252, 253
for events 49
storage 113, 140, 148, 151, 159, 174, 220, 230, 250, 251
swinging door 38, 158, 220
time 110, 141, 148, 151, 153, 155, 172, 184, 220, 228, 253
value 38, 157, 220, 252
Deadband column 45
DeadbandType argument 140, 148, 151, 159, 174, 230
DeadbandType column 113, 118
DefaultStorageRate column 118
DefaultStorageType column 118
DefaultTagRate argument 176
DefaultTagRate column 46
DeleteReplicationServerSystemTags stored procedure 169
Description argument 155, 165, 171, 183, 201, 203, 206, 227, 233, 235, 239, 253
Description column 31, 40, 41, 43, 59, 62, 69, 70, 71, 76, 83, 89, 91, 101, 104, 107, 108, 109, 116, 133, 273, 285, 287
DestinationTagID column 87, 132
DestinationTagName column 87, 131
DetectDateTime argument 180
DetectDateTime column 48
detection time 48, 126, 127, 130, 180
DetectionTime column 126, 127, 130
detector scripts 50, 185
detector string
selecting 170
DetectorClassName column 43
detectors
See event detectors
DetectorString argument 185
DetectorString column 50
DetectorType table 22, 43
DetectorTypeKey argument 185
DetectorTypeKey column 43, 49
DetectValue argument 178
deviation 41, 44
Deviation table 24, 44
discrete tags 45
deleting 171
inserting 171, 292
selecting 175
updating 175, 292
DiscreteHistory table 248, 280
DiscreteHistory view 269
DiscreteLive table 280
DiscreteLive view 269
DiscreteSnapshot table 22, 45
DiscreteTag table 24, 45
DiscreteWideHistory table 248, 249, 281
disk space
event history 224
document conventions 14
double-byte 103, 229
DoubleByte argument 229
DoubleByte column 103

DuplicateChar column 62
Duration argument 233
Duration column 107, 271
dynamic configuration
  committing changes 163, 164, 255
DynDailyAvg view 270
DynDailySum view 270
DynHourlyAvg view 270
DynHourlyMax view 270
DynHourlyMin view 270
DynHourlySum view 270

E

EarliestExecutionDateTimeUtc column 86, 131
Edge argument 178, 180, 185
Edge column 48, 51, 126, 127, 130
edge detection 48, 51, 53, 120, 126, 127, 130, 178, 180, 185
Editable column 109, 273
EditorClassName column 31, 43
Enabled argument 199
Enabled column 67
End argument 224
EndDateTime column 32, 91
EndTime argument 161, 179, 180, 181, 231, 252
engineering units 36, 37, 46, 54, 97, 121, 140, 150, 151, 156
  deleting 176
  inserting 176
  selecting 177
  updating 178
EngineeringUnit table 24, 46
error messages 47, 74, 274
  See also system messages
ErrorCode column 47
ErrorLog table 25, 47
EUKey argument 156, 176, 177, 178
EUKey column 36, 37, 46
event
  SQL scripts 91
event actions 31
  snapshot 90
Event column 126, 127, 129
event detection 178
event detectors 43
event history 179, 180
  deleting events 167
  deleting summaries 167
  managing 224
event tables
  about 22
event tags 49, 51
  deleting 182
  inserting 183
  returning details for 182
  selecting 186
  updating 187
EventArchived column 59
EventHistory table 22, 48
EventLogKey argument 181
EventLogKey column 31, 45, 48, 102
events 48
  See also event actions, event detectors, summaries
EventTag column 272
EventTag table 22, 24, 49
EventTagName argument 181, 231, 233
EventTagName column 90
EventTagPendingDelete table 22, 51
EventTime argument 181
EventTime column 126, 127, 130
ExecuteState column 86, 131
ExecutionMode column 43
ExeName argument 203
ExeName column 70
ExeType argument 202
ExeType column 69
extended stored procedures
  about 246
  history 247
  retrieving list 250
  system 255
  utility 249
extension tables 17
  views for 125

F

faaCheckLicenseViolation function 261
faaContainedName function 262
faaGetHierarchicalAttributeNames function 262
faaGetHistorianTagNames function 262
faaGetLocalizedText function 263

faaLicensedTagDetails function 263
faaLicensedTagTotal function 264
faaObjectTagName function 264
faaTagsInLicenseViolation function 265
faaTZgetdate function 266
faaUser_ID function 266
FALSE 74, 209
FileChunkSize argument 199
FileChunkSize column 68
FileName column 47, 275
FilterStr argument 212, 216, 237, 238
First column 33
FirstDateTime column 33
FixedStorageRate column 62
FKey argument 237
ForwardingDelay argument 200
ForwardingDelay column 68
frequency 51, 107, 114, 241
Frequency column 51, 107
Frequency table 22, 51
FrequencyID argument 241
FrequencyID column 51, 114
FromDate argument 196
FromDate column 59
function 261
functions
  faaGetHierarchicalAttributeNames function 262
  faaGetHistorianTagNames function 262
fww_CheckLicenseViolation function 302
fww_GetLocalizedText function 302
fww_InSQLgetdate function 302
fww_LicensedTagDetails function 302
fww_LicensedTagTotal function 302
fww_TagsInLicenseViolation function 302

G

gid column 119
GroupAbbreviation argument 142
GroupAbbreviation column 81, 134
GroupDefaultSummaryReplicationNamingScheme column 134
GroupID column 282, 287
grouping tables
  about 23
GroupTagList table 282

H

hierarchical attribute name 262
hierarchical name 262
HistorianSysObjects table 23
history 95, 119
history blocks 17, 59
  archiving 249
  creating 250
  rescanning 250
  selecting 196
history data
  extended stored procedures 247
History table 18, 61
  format 19
history tables 17
History view 125
HistoryArchived column 59
HistoryBlock table 18, 59
HistoryBlock view 125
Host column 47, 275

I

I/O Server types 70
  deleting 203
  inserting 203
  selecting 204
I/O Servers 68
  data processing 117, 221, 244
  data processing interval 117, 221, 244
  deleting 201
  failover 69, 202
  inserting 201
  selecting 203
  updating 204, 205
Icon column 285
ID column 40
id column 61, 76, 274
IDASs 64
  autonomous startup timeout 67, 199
  buffers 67, 199
  connection timeout 68, 200
  deleting 197
  failover 64, 199
  inserting 197
  primary 64, 199
  selecting 200
  updating 200

IdleDuration argument 221, 244
IdleDuration column 117
ImageTime argument 223
ImageTime column 90
Import column 118
ImportAllTags column 62
importing
  data dictionary 62
  InTouch node 63, 117
ImportMemoryTags column 62
ImportPlantTags column 62
ImportRoute column 63
ImportSystemTags column 62
InInSQL column 63
InitializationStatus argument 202
InitializationStatus column 69
InitialValue argument 157, 173, 228
InitialValue column 279
InSQLSysObjects table 62, 288
IntegerSize argument 139, 157
IntegerSize column 38
Integral column 35
integral divisor 46, 140, 150, 177
IntegralDivisor argument 140, 150, 177
IntegralDivisor column 46
interpolation 38, 56, 123, 158
InterpolationType argument 158
InterpolationType column 38
IntervalReplicationSchedule table 61
InTouch integration tables
  about 23
InTouch nodes 62, 63
  redirecting data acquisition to 217
  tag list 196
InTouchNode table 23, 62
InTouchNodeKey argument 217
InTouchSpecific table 23, 63
IODriver table 25, 64
IODriverKey argument 197, 200, 202
IODriverKey column 64, 68, 118
IOServer table 25, 68
IOServerKey argument 139, 147, 150, 152, 157, 173, 201, 203, 205, 217, 228, 243
IOServerKey column 68, 109, 116
IOServerType table 25, 70
ItemName argument 153, 155, 172, 227
ItemName column 110

K

Key argument 219, 240

L

LanguageID column 74
Last column 34
LastDateTime column 34
late data 116, 117, 221, 243, 244
LateData argument 221, 243
LateData column 116
licensing 261, 263, 264, 265
limit 41, 70
  for summaries 108, 235
limit names 71, 207
  deleting 206
  inserting 207
  selecting 207
  updating 207
Limit table 24, 70
limit type 70, 206
LimitName table 24, 71
LimitNameKey argument 206, 207
LimitNameKey column 71
limits 71, 206
  deleting 205
  for a storage location 100, 225
  for summaries 108, 235
  priority 71, 206
  selecting 208
  updating 208
LimitType argument 206
LimitType column 70
Line column 47, 275
linear scaling 37, 156
linked server 270
List argument 222
Live table 18, 72
  format 22
Live view 125
locale ID 74, 263
localization 74, 263
LocalizedText column 74, 274
LocalizedText table 25, 74, 263
Logged argument 184
Logged column 50

LogKey column 104, 105
LowerLimit argument 235
LowerLimit column 108

M

MachineName column 62
major deviation 44
MajorChecked column 44
MajorDeviation column 44
MajorPriority column 44
ManualAnalogHistory table 282
ManualDiscreteHistory table 283
ManualStringHistory table 283
ManyOtherTags column 96, 120, 277, 281, 286
MaxAgeThreshold argument 226
MaxAgeThreshold column 100
MaxDateTime column 35
MaxEU argument 140, 151, 156
MaxEU column 36, 37
Maximum column 34
MaximumStates column 87, 132
MaxLength argument 228
MaxLength column 103
MaxMBSize argument 225
MaxMBSize column 100
MaxRaw argument 140, 151, 156
MaxRaw column 37
MaxRowCount argument 252
MDAS
  stored procedures 292, 293
message pairs 74
  deleting 208
  inserting 209
  selecting 209
  updating 209
Message table 24, 74
Message0 argument 209
Message0 column 74
Message1 argument 209
Message1 column 74
MessageKey argument 173, 208, 209
MessageKey column 45, 74
messages
  error 47, 74, 274
  status 74
  system 274
millisecond resolution 135
MinDateTime column 34
MinEU argument 140, 150, 156
MinEU column 36, 37
Minimum column 34
MinMBThreshold argument 198, 226
MinMBThreshold column 66, 100
minor deviation 44
MinorChecked column 44
MinorDeviation column 44
MinorPriority column 44
MinRaw argument 140, 151, 156
MinRaw column 36, 37
MinSFDuration column 85, 133
modification tracking 40, 61, 75, 128, 274
  returning list 166
  status for 210
  tables 23
Modified column 105
ModLogColumn table 23, 75
ModLogTable table 23, 75
ModStartDateTimeUtc column 85, 131
ModStopDateTimeUtc column 86, 131
ModTableKey column 75
ModType column 76, 128
ModuleID column 47, 275

N

Name argument 207, 210, 213, 216, 238, 239, 243
Name column 31, 43, 61, 71, 78, 79, 108, 116, 272, 273, 274, 285
NamedSystemParameter view 272
NameKey argument 211, 212, 213, 214, 215, 216
NameKey column 77, 78, 273
namespace 77, 78, 284
  ArchestrA 29
  tables 23
NameSpaceIcons table 284
NewValue column 75, 128
NodeKey argument 196
NodeKey column 62, 63, 117
NumericValue column 272

O

object 29
  ArchestrA 30
ObjectKey column 29, 30, 41

ObjName argument 292
Offset column 116
OldValue column 75, 129
OnLine column 59
OPC 52, 72, 76
OPC quality 56, 76, 98, 123
OPCQuality column 33, 52, 72, 76, 92, 276, 277, 280, 281, 285, 286
OPCQualityMap table 25, 76
operating system
  for I/O Servers 204
OperationEnd column 106
OperationKey argument 232, 234, 235, 236
OperationKey column 105, 107, 108
operations
  See summary operations
OperationStart column 106
Operator argument 178
OpKey argument 223
OrderBy argument 182, 231
OriginalName column 63, 79

P

parameter 108
Parameter column 47, 274
ParentKey argument 210, 213
ParentKey column 28, 30, 78, 79, 273
Password column 83, 133
Path argument 202, 225
Path column 62, 69, 100
PercentGood column 33, 57
Period column 61
Periodicity argument 241
Periodicity column 115
PKey argument 194, 236, 237, 238
Pkey argument 212, 216
PKey column 273
Platform argument 204
PostDetectorDelay argument 184
PostDetectorDelay column 50
PrefixOrSuffix column 62
Priority argument 185, 206
Priority column 51, 71, 80
private namespaces 23, 77
  adding a group 210
  adding a leaf 211
  deleting a group 211
  deleting a leaf 212
  expanding 212
  selecting object in 213
  updating a group 213
PrivateGroupTag table 23, 77
PrivateNameSpace table 23, 77
processing interval 117, 221, 244
ProcessingInterval argument 244
ProcessingInterval column 117
ProcInterval argument 221
protocols 69, 118, 202
ProtocolType argument 202
ProtocolType column 69, 118
Provider column 279
public namespaces 23, 78
  adding a group 213
  adding a leaf 214
  deleting a group 215
  deleting a leaf 215
  expanding 216
  selecting object in 216
  updating group 216
PublicGroupTag table 23, 78
PublicNameSpace table 23, 78

Q

quality 32, 45, 52, 72, 76, 102, 104, 127, 128, 130, 271, 278, 282, 283, 284
  percentage 57
Quality column 32, 45, 52, 72, 102, 104, 127, 128, 130, 271, 276, 277, 278, 280, 281, 282, 283, 284, 285, 286
quality detail 79
quality rule 56, 98, 123
QualityDetail column 32, 45, 52, 72, 79, 102, 127, 128, 130, 276, 277, 278, 280, 281, 283, 284, 285, 286
QualityMap table 25, 79
QualityString column 79

R

rate of change 41, 80
RateDB argument 220
RateDeadband argument 158
RateDeadband column 38, 118
RateOfChange table 24, 80
raw value 36, 37, 140, 151, 156
RawType argument 157

RawType column 37
reconfiguration
  See dynamic configuration
referential integrity 27
RegistryName column 116
remote table 17
  See also extension tables
replication tables 24
ReplicationGroup table 80
ReplicationGroupKey argument 142
ReplicationGroupKey column 80, 87, 132
ReplicationGroupName argument 142
ReplicationGroupName column 80, 134
ReplicationSchedule table 82
ReplicationScheduleAbbreviation column 82
ReplicationScheduleKey column 42, 61, 81, 82, 134
ReplicationScheduleName argument 142
ReplicationScheduleName column 82
ReplicationScheduleType column 82
ReplicationScheduleType table 82
ReplicationScheduleTypeKey column 82
ReplicationScheduleTypeName column 82
ReplicationServer table 83
ReplicationServerKey column 80, 83, 87, 133
ReplicationServerName argument 142
ReplicationServerName column 83, 130
ReplicationSyncRequest table 85
ReplicationSyncRequestInfo view 130
ReplicationSyncRequestKey column 85, 131
ReplicationTag table 86
ReplicationTagEntity table 86
ReplicationTagEntityKey column 85, 131
ReplicationType table 89
ReplicationTypeKey argument 142
ReplicationTypeKey column 81, 89, 134
ReplicationTypeName column 89
RequestInitialData column 117
RequestVersion column 85, 131
resolution 53, 96, 107, 120, 179, 233, 252, 272
Resolution argument 179, 233, 252
Resolution column 107, 272
retrieval cycles 55, 97, 122
Retriever column 279
Revision argument 204
Revision column 70
rollover value 39, 158
RolloverValue argument 158
RolloverValue column 39
row count 53, 96, 120
RowKey column 76, 128
RunTimeDay argument 241
RunTimeDay column 115
RunTimeHour argument 242
RunTimeHour column 115
RunTimeMin argument 242
RunTimeMin column 115

S

samples
  active image 218
Samples argument 218, 245
SamplesInActiveImage argument 158, 174, 229
SamplesInActiveImage column 111
scaling 37, 156
Scaling argument 156
Scaling column 37
scan rate
  for event tags 49, 184
ScanRate argument 184
ScanRate column 49
Script column 91
script template 91
security
  levels 119
Sequence column 60
server list 89
ServerDefaultSimpleReplicationNaming Scheme column 134
ServerDefaultSummaryReplicationNamingScheme column 134
ServerKey argument 211, 214
ServerKey column 89, 113
ServerList table 23, 89
ServerTimeStamp argument 158, 174, 219, 221, 229
ServerTimeStamp column 112
Set argument 222
SFFreeSpace column 83, 133
SFPath column 83, 134
SignedInteger argument 139, 157

SignedInteger column 38
SimpleReplicationNamingScheme column 84
snapshot 31, 45, 90, 102, 126, 127, 129
snapshot actions 90, 127
snapshot files 89, 222, 223
SnapshotDetail table 25, 89
snapshots 181
SnapshotSize argument 223
SnapshotSize column 90
SnapshotTag table 22, 90
SnapshotTagKey column 31, 45, 90, 102
SortOrder argument 225
SortOrder column 99
SourceName column 117
SourceServer column 36, 57, 73, 86, 94
SourceTag column 35, 57, 73, 86, 94
SourceTagName column 87, 130
SourceType column 107
sp_help stored procedure 27
SQL script 91
SQLTemplate table 22, 91, 103, 104
Start argument 223
StartDateTime argument 241
StartDateTime column 32, 57, 91, 98, 115, 124
StartTime argument 161, 179, 180, 181, 231, 252
StateCount column 92
states for values 57, 98
StateSummaryHistory table 18, 91
StateSummaryHistory view 126
StateTime column 57
StateTimeAvg column 93
StateTimeAvgContained column 94
StateTimeMax column 93
StateTimeMaxContained column 93
StateTimeMin column 92
StateTimeMinContained column 93
StateTimePercent column 94
StateTimePercentContained column 94
StateTimeTotal column 94
StateTimeTotalContained column 94
StateWideHistory table 18, 95
Status argument 184
Status column 30, 41, 50, 67, 70, 81, 85, 88, 90, 100, 106, 109, 112, 116, 133
StdDev column 35
storage
  deadband 113, 140, 148, 151, 159, 174, 220, 230
  setting deadband 250, 251
  tags 221
  See also history blocks, storage locations
Storage argument 195
storage locations 99
  path 100, 225
  selecting 224
  updating 225
storage node 101
storage rate 110, 153, 155, 172, 220, 227
storage rules
  tags 219
storage type 110, 140, 148, 150, 153, 155, 172, 220, 227
StorageAreaType column 60
StorageLocation table 25, 99
StorageNode table 25, 101
StorageNodeKey argument 139, 147, 150, 152, 157, 197, 201, 224, 225, 243
StorageNodeKey column 59, 64, 68, 99, 101, 109, 116
StorageRate argument 153, 155, 172, 220, 227
StorageRate column 110
StorageSize argument 222, 223
StorageSize column 89
StorageType argument 140, 148, 150, 153, 155, 172, 220, 224, 225, 227
StorageType column 99, 110
store-and-forward 65, 66, 198
  data chunk size 68, 199
  duration 67, 199
  interval 68, 200
  starting data processing 251
stored procedures
  about 137
  backward compatibility 288
  creating 258
StoreForwardDuration argument 199
StoreForwardDuration column 67
StoreForwardMode argument 198
StoreForwardMode column 65
StoreForwardPath argument 198
StoreForwardPath column 66

string tags 102, 103
  deleting 226
  inserting 227, 293
  selecting 230
  updating 230, 293
StringHistory table 285
StringHistory view 270
StringLive table 286
StringLive view 270
StringSnapshot table 22, 102
StringTag table 24, 103
StringValue column 272
StringWideHistory table 286
StructureAttributes table 103
StructureID column 103, 104
StructureTag table 104
StructureType table 104
StructureTypeName column 104
SumDateCalcType column 105
SumDateDuration column 106
SumDateResolution column 106
SumDateTime argument 231
SumDateTimeStamp column 105
summaries
  backward compatibility views 270
  deleting from history 167
summary actions
  inserting 231
summary calculation 40
summary data 104, 270, 271
summary date 105, 271
summary history
  managing 224
summary information
  deleting 234
  inserting 235
  selecting 236
  updating 236
summary operations 51, 105, 107, 108
  deleting 232
  inserting 232
  retrieving details 231
  returning details 231
  selecting 233
  updating 234
summary tag 108
SummaryArchived column 59
SummaryData table 22, 104
SummaryDate column 105, 271
SummaryHistory table 22, 105
SummaryOperation table 22, 107
SummaryReplicationNamingScheme argument 142
SummaryReplicationNamingScheme column 81, 84
SummaryTagList table 22, 108
SumVarKey argument 234, 236
SumVarKey column 104, 108
swinging door deadband 38, 158, 220
system configuration tables
  about 25
system messages 47, 74, 274
system namespace 23, 273
  expanding 236, 237, 238
system parameters 108, 238, 239, 272
  selecting 238
  updating 238
system startup 163
SystemNameSpace view 273
SystemParameter table 25, 108

T

Table column 128
tables
  about 27
  accessing information 27
  categories 17
  referential integrity 27
tag count 261, 264, 265
tag definitions
  tables 24
tag groups 287
Tag table 22, 23, 24, 25, 109
Tag1 column 96, 120
Tag2 column 96, 120
TagA1 column 277
TagA2 column 277
TagD1 column 281
TagD2 column 281
TagGroup table 287
TagId column 113
TagKey argument 245
TagList argument 154, 161, 170, 180, 181, 182, 226, 231, 240, 241, 289, 290
TagN argument 253

TagName argument 138, 147, 149, 152, 155, 160, 170, 171, 178, 179, 183, 188, 193, 205, 208, 218, 227, 232, 235, 236, 241, 242
TagName column 32, 36, 37, 39, 44, 45, 48, 49, 52, 63, 70, 72, 80, 86, 90, 91, 103, 107, 108, 109, 114, 126, 128, 130, 270, 271, 276, 277, 278, 279, 280, 282, 283, 284, 285, 286, 288
tagname database
  See data dictionary
Tagname.X file 62, 63
TagRef table 23, 24, 113
tags 109, 113
  deleting 170
  retrieving definition 240
  retrieving types 241
  setting storage rules 219
  storage for 221
TagS1 column 286
TagS2 column 286
TagType argument 153, 188, 240
TagType column 90, 110, 114
TagType table 114
TagTypeKey column 114
TagTypeName column 114
Target column 44
TCPPort column 83, 134
TemplateKey column 91, 104
TextKey column 74
threads 50, 184
threshold
  for storage 100, 226
  IDAS 66, 198
ThresholdTime argument 223
ThresholdTime column 90
time deadband 110, 141, 148, 151, 153, 155, 172, 184, 220, 228, 253
  for storage 250
time detectors 114, 115
  inserting 241
  selecting 242
  updating 242
time interval
  for event tags 49, 184
time zone 55, 60, 73, 97, 115, 122, 256, 266
TimeBase column 80
TimeDB argument 220
TimeDeadband argument 141, 148, 151, 155, 172, 184, 228, 253
TimeDeadband column 49, 110, 118
TimeDetectorDetail table 22, 114
TimeDetectorDetailKey column 114
TimeDetectorDetailPendingDelete table 22, 115
time-in-state 57
TimeOfDay column 43
timeout 116, 243
  IDAS autonomous startup 67, 199
  IDAS connection 68, 200
TimeOut argument 243
TimeOut column 116
TimeStamp argument 233
TimeStamp column 107, 272
timestamp rule 55, 97, 122
timestamping 112, 158, 174, 219, 221, 229
TimeZone column 115
TimeZone table 25, 115
TimeZoneID column 115
TimeZoneOffset column 60
ToDate argument 196
ToDate column 59
TokenType argument 237
topic definition 117
Topic table 24, 25, 116
TopicImportInfo table 23, 117
TopicKey argument 139, 147, 150, 152, 157, 173, 228, 243, 244
TopicKey column 109, 116
TopicName argument 219
TopicName column 117
topics 116, 117, 219, 243
  deleting 243
  inserting 243
  selecting 244
  updating 244
TotalCount column 47, 275
tracking
  See modification tracking
Triggerval column 282
TRUE 74, 209
Type argument 210, 213, 214, 217, 219, 221
Type column 29, 30, 40, 47, 61, 77, 79, 91, 272, 273, 274, 279, 284, 287
TypeInfo column 63

U

uid column 119
Unit argument 176
Unit column 46, 61
unit of measure 46, 140, 150, 176
UpperLimit argument 235
UpperLimit column 108
user 246
User column 129
user ID 119, 266
UserDetail table 23, 25, 118
UserKey argument 160
UserKey column 39, 76, 77, 119, 287
UserName column 76, 83, 119, 134
users 76, 119, 129
UseThreadPool argument 184
UseThreadPool column 50
UTC 55, 60, 73, 97, 122

V

v_AnalogHistory view 269
v_AnalogLive view 269
v_DiscreteHistory view 269
v_DiscreteLive view 269
v_ErrorLog view 274
v_EventSnapshot view 126
v_EventStringSnapshot view 127
v_History view 270
v_HistoryBlock view 270
v_Live view 270
v_ModTracking view 128
v_SnapshotData view 129
v_StringHistory view 270
v_StringLive view 270
v_SummaryData view 271
Value argument 161, 206, 239
Value column 31, 39, 45, 52, 71, 72, 80, 91, 102, 104, 108, 127, 128, 130, 271, 276, 277, 280, 281, 282, 283, 284, 285, 286
value deadband 38, 157, 220, 252
  storage 251
ValueCount column 35
ValueDB argument 220
ValueDeadBand argument 252
ValueDeadband argument 157
ValueDeadband column 38, 118
version
  for data 55, 97, 122
  Tag table 113
Version column 28, 60
views
  about 125
  backward compatibility 269
vValue column 52, 72, 92, 96

W

WideHistory table 18, 119
  format 20
WideTableDictionary table 287
Windows login
  See logins
ww_ActionStringSelect stored procedure 293
ww_AddTag stored procedure 293
ww_AnalogDetail stored procedure 293
ww_AnalogTagDelete stored procedure 293
ww_AnalogTagInsert stored procedure 293
ww_AnalogTagSelect stored procedure 293
ww_AnalogTagUpdate stored procedure 293
ww_Annotation stored procedure 293
ww_AnnotationDelete stored procedure 293
ww_AnnotationInsert stored procedure 293
ww_AnnotationSelect stored procedure 293
ww_AnnotationUpdate stored procedure 293
ww_CheckClientVersion stored procedure 290, 293
ww_CheckWhichDb stored procedure 291, 293
ww_CleanupAfterCommit stored procedure 293
ww_CommitChanges stored procedure 293
ww_CommitChangesAtStartup stored procedure 294, 298
ww_ContextDelete stored procedure 294, 298
ww_ContextInsert stored procedure 294, 298

ww_ContextSelect stored procedure 294, 298
ww_ContextUpdate stored procedure 294, 298
ww_DBChangesPending stored procedure 294, 298
ww_dbCheck stored procedure 291, 294, 298
ww_DBConfig stored procedure 291, 294, 298
ww_DeleteOlderEvents stored procedure 294, 298
ww_DeleteOlderSummaries stored procedure 294, 298
ww_DeleteTag stored procedure 294, 298
ww_DetectorStringSelect stored procedure 294, 298
ww_DiscreteDetail stored procedure 294, 298
ww_DiscreteTagDelete stored procedure 294, 298
ww_DiscreteTagInsert stored procedure 294, 298
ww_DiscreteTagSelect stored procedure 294, 298
ww_DiscreteTagUpdate stored procedure 294, 298
ww_EngineeringUnitDelete stored procedure 294, 298
ww_EngineeringUnitInsert stored procedure 294, 298
ww_EngineeringUnitSelect stored procedure 294, 298
ww_EngineeringUnitUpdate stored procedure 294, 298
ww_EventDetection stored procedure 294, 298
ww_EventHistory stored procedure 294, 298
ww_EventHistoryInsert stored procedure 294, 298
ww_EventSnapshot stored procedure 294, 298
ww_EventSnapshotInsert stored procedure 294, 298
ww_EventTagDelete stored procedure 294, 298
ww_EventTagDetail stored procedure 294, 298
ww_EventTagInsert stored procedure 294, 298
ww_EventTagSelect stored procedure 294, 298
ww_EventTagSelectAll stored procedure 295, 299
ww_EventTagSelectDeleted stored procedure 295, 299
ww_EventTagSelectDisabled stored procedure 295, 299
ww_EventTagSelectInserted stored procedure 295, 299
ww_EventTagSelectUpdated stored procedure 295, 299
ww_EventTagUpdate stored procedure 295, 299
ww_GetDbRevision stored procedure 295, 299
ww_GetLastTagKey stored procedure 295, 299
ww_HistoryBlockSelect stored procedure 295, 299
ww_InSQLConfigNSExpand stored procedure 295, 299
ww_InSQLNSExpand stored procedure 295, 299
ww_InSQLStatusSelect stored procedure 295, 299
ww_InSQLStatusSet stored procedure 295, 299
ww_InTouchNodeTagList stored procedure 295, 299
ww_IODriverDelete stored procedure 295, 299
ww_IODriverInsert stored procedure 295, 299
ww_IODriverSelect stored procedure 295, 299
ww_IODriverUpdate stored procedure 295, 299
ww_IOServerDelete stored procedure 295, 299
ww_IOServerInsert stored procedure 295, 299
ww_IOServerSelect stored procedure 295, 299
ww_IOServerTypeDelete stored procedure 295, 299
ww_IOServerTypeInsert stored procedure 295, 299
ww_IOServerTypeSelect stored procedure 295, 299

ww_IOServerTypeUpdate stored procedure 295, 299
ww_IOServerUpdate stored procedure 295, 299
ww_LimitDelete stored procedure 295, 299
ww_LimitInsert stored procedure 295, 299
ww_LimitNameDelete stored procedure 295, 299
ww_LimitNameInsert stored procedure 295, 299
ww_LimitNameSelect stored procedure 296, 300
ww_LimitNameUpdate stored procedure 296, 300
ww_LimitSelect stored procedure 296, 300
ww_LimitUpdate stored procedure 296, 300
ww_LoadInSQLProcedureBody stored procedure 292, 296, 300
ww_MDASAnalogTagInsert stored procedure 292
ww_MDASAnalogTagUpdate stored procedure 292
ww_MDASDiscreteTagInsert stored procedure 292
ww_MDASDiscreteTagUpdate stored procedure 292
ww_MDASStringTagInsert stored procedure 293
ww_MDASStringTagUpdate stored procedure 293
ww_MessageDelete stored procedure 296, 300
ww_MessageInsert stored procedure 296, 300
ww_MessageSelect stored procedure 296, 300
ww_MessageUpdate stored procedure 296, 300
ww_ModLogStatus stored procedure 296, 300
ww_PrivateNSAddGroup stored procedure 296, 300
ww_PrivateNSAddLeaf stored procedure 296, 300
ww_PrivateNSDeleteGroup stored procedure 296, 300
ww_PrivateNSDeleteLeaf stored procedure 296, 300
ww_PrivateNSExpand stored procedure 296, 300
ww_PrivateNSSelect stored procedure 296, 300
ww_PrivateNSUpdateGroup stored procedure 296, 300
ww_PublicNSAddGroup stored procedure 296, 300
ww_PublicNSAddLeaf stored procedure 296, 300
ww_PublicNSDeleteGroup stored procedure 296, 300
ww_PublicNSDeleteLeaf stored procedure 296, 300
ww_PublicNSExpand stored procedure 296, 300
ww_PublicNSSelect stored procedure 296, 300
ww_PublicNSUpdateGroup stored procedure 296, 300
ww_RedirectToInTouch stored procedure 296, 300
ww_SetAISamples stored procedure 296, 300
ww_SetCalculatedAISamples stored procedure 296, 300
ww_SetStorageRule stored procedure 296, 300
ww_SetTagStorage stored procedure 296, 300
ww_SnapshotDetailSelect stored procedure 296, 300
ww_SnapshotDetailUpdate stored procedure 297, 301
ww_SnapToSummary stored procedure 297, 301
ww_SpaceManager stored procedure 297, 301
ww_StorageLocationSelect stored procedure 297, 301
ww_StorageLocationUpdate stored procedure 297, 301
ww_StringDetail stored procedure 297, 301
ww_StringTagDelete stored procedure 297, 301
ww_StringTagInsert stored procedure 297, 301

ww_StringTagSelect stored procedure 297, 301
ww_StringTagUpdate stored procedure 297, 301
ww_SummaryActionInsert stored procedure 297, 301
ww_SummaryDetail stored procedure 297, 301
ww_SummaryOperationDelete stored procedure 297, 301
ww_SummaryOperationInsert stored procedure 297, 301
ww_SummaryOperationSelect stored procedure 297, 301
ww_SummaryOperationUpdate stored procedure 297, 301
ww_SummaryTagListDelete stored procedure 297, 301
ww_SummaryTagListInsert stored procedure 297, 301
ww_SummaryTagListSelect stored procedure 297, 301
ww_SummaryTagListUpdate stored procedure 297, 301
ww_SystemConfigNSExpand stored procedure 297, 301
ww_SystemNSExpand stored procedure 297, 301
ww_SystemNSExpand2 stored procedure 297, 301
ww_SystemParameterSelect stored procedure 297, 301
ww_SystemParameterUpdate stored procedure 297, 301
ww_TagConfig stored procedure 297, 301
ww_TagConfigModified stored procedure 297, 301
ww_TagConfigSelect stored procedure 297, 301
ww_TagInfo stored procedure 297, 301
ww_TagType stored procedure 297, 301
ww_TimeDetectorDetailInsert stored procedure 302
ww_TimeDetectorDetailSelect stored procedure 302
ww_TimeDetectorDetailUpdate stored procedure 302
ww_TopicDelete stored procedure 302
ww_TopicInsert stored procedure 302
ww_TopicSelect stored procedure 302
ww_TopicUpdate stored procedure 302
ww_UpdateCalculatedAISamples stored procedure 302
ww_UserAccessLevelSelect stored procedure 302
ww_UserDetailUpdate stored procedure 302
wwAdmin login 289
wwCycleCount column 36, 55, 95, 97, 122, 276, 278, 280, 282, 285, 287
wwdbo user 289
wwDomainTagKey column 29, 77, 78, 113, 282
wwEdgeDetection column 53, 120, 276, 277, 280, 281, 285, 286
wwFilter column 58, 124
wwInterpolationType column 56, 123, 276, 278
wwMaxStates column 58, 95
wwParameters column 57, 73, 98, 124, 276, 277, 278, 280, 281, 282, 285, 286, 287
wwPower login 289
wwQualityRule column 56, 98, 123, 276, 278, 280, 285
wwResolution column 36, 53, 95, 96, 120, 276, 277, 280, 281, 285, 286
wwRetrievalMode column 36, 54, 72, 95, 96, 121, 276, 277, 278, 280, 281, 285, 286, 287
wwRowCount column 53, 96, 120, 276, 277, 280, 281, 285, 286
wwStateCalc column 57, 98
wwTagKey argument 141, 148, 152, 154, 159, 171, 175, 182, 186, 187, 211, 212, 214, 215, 226, 230
wwTagKey column 36, 52, 72, 95, 109, 113, 276, 277, 280, 281, 283, 284, 285, 286
wwTimeDeadband column 54, 72, 96, 121, 276, 277, 278, 280, 281, 285, 286, 287
wwTimeStampRule column 55, 97, 122, 276, 278, 280, 282, 285, 287
wwTimeZone argument 256
wwTimeZone column 36, 55, 60, 73, 95, 97, 122, 276, 277, 278, 280, 281, 282, 285, 286, 287
wwUser login 289
wwValueDeadband column 54, 72, 97, 121, 276, 277, 278
wwValueSelector column 58, 73, 124

wwVersion column 36, 55, 95, 97, 122, 276, 278, 280, 282, 285, 287

X

xp_AllowCommit extended stored procedure 255
xp_AnalogHistory extended stored procedure 247
xp_AnalogHistoryDelta extended stored procedure 247
xp_AnalogWideHistory extended stored procedure 248
xp_AnalogWideHistoryDelta extended stored procedure 248
xp_DiscreteHistory extended stored procedure 248
xp_DiscreteHistoryDelta extended stored procedure 248
xp_DiscreteWideHistory extended stored procedure 248
xp_DiscreteWideHistoryDelta extended stored procedure 249
xp_DiskCopy extended stored procedure 249
xp_NewHistoryBlock extended stored procedure 250
xp_ProcList extended stored procedure 250
xp_RescanHistoryBlocks extended stored procedures 250
xp_SetStorageTimeDeadband extended stored procedure 250
xp_SetStorageValueDeadband extended stored procedure 251
xp_SetStoreForwardEvent stored procedure 251
xp_TZGetDate extended stored procedure 256