
GRID AUTOMATION PRODUCTS

MicroSCADA X SYS600 10.2

Application Objects





Document ID: 1MRK 511 467-UEN
Issued: March 2021
Revision: A
Product version: 10.2

© 2021 Hitachi Power Grids. All rights reserved.

Table of contents

Section 1	Copyrights.....	19
Section 2	About this manual.....	21
2.1	Use of symbols.....	21
2.2	Intended audience.....	21
2.3	Related documents.....	21
2.4	Document conventions.....	22
2.5	Document revisions.....	22
Section 3	Introduction.....	23
3.1	About this section.....	23
3.2	SYS600.....	23
3.3	Applications.....	23
3.4	Application objects.....	24
3.5	Attributes.....	25
3.6	Defining application objects.....	26
3.7	Databases.....	26
Section 4	Object handling.....	29
4.1	Defining application objects	29
4.1.1	Using definition tools.....	29
4.1.2	Using LIB 500.....	29
4.1.3	Using Connectivity Packages or SCL files.....	29
4.1.4	Defining objects with SCIL.....	30
4.2	Using application objects in SCIL.....	30
4.2.1	General.....	30
4.2.2	Object notation.....	30
4.2.3	Object attribute notation.....	30
4.2.4	Name.....	31
4.2.5	Application.....	31
4.2.6	Type.....	31
4.2.7	Attribute.....	32
4.2.8	Index.....	32
4.2.9	Using object notations.....	33
4.2.10	Attribute access level.....	33
4.3	Some SCIL commands.....	33
4.3.1	General.....	33
4.3.2	Setting attribute values.....	34
4.3.2.1	#SET attribute {= expression}.....	34
4.3.3	Executing objects.....	34
4.3.3.1	#EXEC object {(variable_list)}.....	34
4.3.4	Event object handling.....	34

4.3.4.1	#ON event_object [statement].....	34
4.3.5	Updating process database.....	34
4.3.5.1	#GET object.....	34
4.3.6	Process queries.....	34
4.3.6.1	APPLICATION_OBJECT_LIST, APPLICATION_OBJECT_ATTRIBUTES, APPLICATION_ALARM_LIST, HISTORY_DATABASE_MANAGER.....	34
4.3.7	Creating objects.....	34
4.3.7.1	#CREATE object [= expression].....	34
4.3.8	Modifying objects.....	35
4.3.8.1	#MODIFY object = expression.....	35
4.3.9	Deleting objects.....	35
4.3.9.1	#DELETE object.....	35
Section 5	Process objects.....	37
5.1	General.....	37
5.1.1	Use.....	37
5.1.2	Function.....	37
5.1.3	Process object types.....	38
5.1.4	User defined types.....	38
5.1.5	Attributes.....	40
5.1.6	Process object groups.....	42
5.1.7	Storage.....	43
5.1.8	Process object notation.....	43
5.2	Configurable process object attributes.....	44
5.2.1	Basic definition attributes.....	44
5.2.1.1	IX Index.....	44
5.2.1.2	LN Logical Name.....	45
5.2.1.3	PT Process Object Type.....	45
5.2.1.4	ZT Modification Time.....	45
5.2.2	Identification attributes.....	46
5.2.2.1	CX Comment Text	46
5.2.2.2	ES Event Source.....	46
5.2.2.3	IE Identifier Elements.....	46
5.2.2.4	IL Identifier List.....	47
5.2.2.5	OI Object Identifier.....	47
5.2.2.6	ON OPC Item Name.....	48
5.2.2.7	OX Object Text.....	48
5.2.2.8	TX Translated Object Text.....	48
5.2.3	Addresses.....	49
5.2.3.1	IG Item Group.....	49
5.2.3.2	IN Item Name.....	49
5.2.3.3	OA Object Address.....	50
5.2.3.4	OB Object Bit Address.....	51
5.2.3.5	OT Output Type.....	51
5.2.3.6	TI Table Index.....	52
5.2.3.7	UN Unit Number.....	52
5.2.4	Operational state.....	52

5.2.4.1	IU In Use.....	52
5.2.4.2	SS Switch State.....	53
5.2.4.3	SU Substitution State.....	53
5.2.5	Scaling.....	54
5.2.5.1	BC Bit Count	54
5.2.5.2	DP Decimal Places.....	54
5.2.5.3	IR Integer Representation.....	54
5.2.5.4	SC Scaling.....	55
5.2.5.5	SN Scale Name.....	55
5.2.5.6	ST Engineering Unit.....	55
5.2.6	Alarm handling.....	56
5.2.6.1	General.....	56
5.2.6.2	Alarm generation.....	58
5.2.6.3	Alarm handling attributes.....	61
5.2.7	Limit value supervision.....	63
5.2.7.1	HI High Input.....	64
5.2.7.2	HO High Output.....	64
5.2.7.3	HW High Warning.....	64
5.2.7.4	LI Low Input.....	64
5.2.7.5	LO Low Output.....	65
5.2.7.6	LW Low Warning.....	65
5.2.7.7	SZ SCADA Zone Supervision.....	65
5.2.7.8	ZE Zero Deadband Supervision Enabled.....	65
5.2.7.9	ZD Zero Deadband.....	66
5.2.8	Post-processing.....	66
5.2.8.1	General.....	66
5.2.8.2	EH Event Handling.....	67
5.2.8.3	PO Post-processing Based on Object Value.....	68
5.2.8.4	Activation criterion.....	68
5.2.8.5	First update.....	70
5.2.8.6	Snapshot variables.....	70
5.2.9	Event handling.....	71
5.2.9.1	Event channel activation.....	71
5.2.9.2	Event activation.....	71
5.2.9.3	Event handling attributes.....	72
5.2.10	Logging of event history.....	74
5.2.10.1	General.....	74
5.2.10.2	History database.....	75
5.2.10.3	History buffer and log.....	75
5.2.10.4	History configuration attributes.....	76
5.2.11	Historian logging.....	77
5.2.11.1	GN Logging Name.....	77
5.2.11.2	GP Logging Profile.....	77
5.2.12	OPC event generation.....	78
5.2.12.1	General.....	78
5.2.12.2	Functionality.....	78

5.2.12.3	Event sources.....	79
5.2.12.4	Event categories.....	79
5.2.12.5	Simple and tracking events.....	79
5.2.12.6	Conditions.....	80
5.2.12.7	Vendor specific event attributes.....	80
5.2.12.8	Registration and configuration.....	80
5.2.12.9	Engineering.....	80
5.2.13	Printout handling.....	81
5.2.13.1	LD Listing Devices.....	81
5.2.13.2	PA Printout Activation.....	81
5.2.13.3	PF Physical Format.....	82
5.2.13.4	PH Printout on History.....	82
5.2.13.5	PU Printout at First Update.....	82
5.2.14	Network topology.....	82
5.2.14.1	CN Connected Network Objects.....	83
5.2.14.2	NM Network Topology Model.....	83
5.2.14.3	NO Network Object Connection.....	83
5.2.14.4	NS Network Object Subtype.....	83
5.2.14.5	RO Referenced Objects.....	84
5.2.15	Miscellaneous attributes.....	84
5.2.15.1	Counter definition attributes.....	84
5.2.15.2	SCIL application attributes.....	85
5.3	Dynamic process object attributes.....	86
5.3.1	Object value.....	87
5.3.1.1	AI Analog Input.....	87
5.3.1.2	AO Analog Output.....	87
5.3.1.3	BI Binary Input.....	87
5.3.1.4	BO Binary Output.....	88
5.3.1.5	BS Bit Stream.....	88
5.3.1.6	DB Double Binary Indication.....	88
5.3.1.7	DI Digital Input.....	89
5.3.1.8	DO Digital Output.....	89
5.3.1.9	NT Network Object State.....	89
5.3.1.10	OE OPC Event.....	90
5.3.1.11	OV Object Value.....	90
5.3.1.12	PC Pulse Counter.....	91
5.3.1.13	SX State Text.....	91
5.3.1.14	TS Topological State.....	91
5.3.2	Time and validation stamps.....	92
5.3.2.1	CS Control Status.....	92
5.3.2.2	OS Object Status.....	92
5.3.2.3	RM Registration Milliseconds.....	93
5.3.2.4	RQ Registration Qualified Time.....	93
5.3.2.5	RT Registration Time.....	93
5.3.3	Alarm and warning states.....	94
5.3.3.1	AL Alarm.....	94

5.3.3.2	AR	Alarm Receipt.....	94
5.3.3.3	AS	Alarm State.....	94
5.3.3.4	AV	Alarm Severity.....	95
5.3.3.5	AM	Alarm Milliseconds.....	95
5.3.3.6	AQ	Alarm Qualified Time.....	95
5.3.3.7	AT	Alarm Time.....	95
5.3.3.8	AZ	Alarm Zone.....	96
5.3.3.9	KM	Acknowledgement Milliseconds.....	96
5.3.3.10	KQ	Acknowledgement Qualified Time.....	97
5.3.3.11	KT	Acknowledgement Time.....	97
5.3.3.12	WQ	Warning Qualified Time.....	97
5.3.3.13	YM	Alarm On Time Milliseconds.....	97
5.3.3.14	YQ	Alarm On Qualified Time.....	98
5.3.3.15	YT	Alarm On Time.....	98
5.3.4		Blocking attributes.....	98
5.3.4.1	AB	Alarm Blocking.....	98
5.3.4.2	HB	History Blocking.....	99
5.3.4.3	PB	Printout Blocking.....	99
5.3.4.4	UB	Update Blocking.....	99
5.3.4.5	XB	Activation Blocking.....	100
5.3.5		Operation counter attributes.....	100
5.3.5.1	CO	Counter Overflow.....	100
5.3.5.2	CV	Counter Value.....	100
5.3.6		Minimum and maximum values.....	101
5.3.6.1	MM	Minimum Time Milliseconds.....	101
5.3.6.2	MQ	Minimum Qualified Time.....	101
5.3.6.3	MT	Minimum Time.....	101
5.3.6.4	MV	Minimum Value.....	101
5.3.6.5	XM	Maximum Time Milliseconds.....	102
5.3.6.6	XQ	Maximum Qualified Time.....	102
5.3.6.7	XT	Maximum Time.....	102
5.3.6.8	XV	Maximum Value.....	102
5.3.7		Stamps set by the communication system.....	102
5.3.7.1	BL	Blocked.....	103
5.3.7.2	CT	Cause of Transmission.....	103
5.3.7.3	OR	Out of Range.....	104
5.3.7.4	RA	Reserved A.....	104
5.3.7.5	SB	Substituted.....	105
5.3.7.6	TM	Test Mode.....	105
5.3.8		S.P.I.D.E.R. RTU specific attributes.....	105
5.3.8.1	EP	End of Period.....	105
5.3.8.2	OF	Overflow.....	105
5.3.8.3	SE	Selection.....	106
5.3.8.4	SP	Stop Execution.....	106
5.3.9		IEC, REX and DNP specific attributes.....	107
5.3.9.1	OG	Originator Identification.....	107

5.3.9.2	QL Command Qualifier.....	107
5.3.9.3	TY Type Identification.....	108
5.3.10	OPC Event objects.....	108
5.3.10.1	Basic functionality.....	108
5.3.10.2	Full functionality.....	109
5.3.10.3	Undefined event sources.....	109
5.3.10.4	Initial state of condition events.....	109
5.3.10.5	Setting object status by SCIL.....	110
5.3.10.6	Alarm acknowledgement.....	111
5.3.10.7	OPC Event attributes.....	111
5.3.11	File transfer attributes.....	115
5.3.11.1	DC Directory Contents.....	116
5.3.11.2	FF File Function.....	116
5.3.11.3	FN File Name.....	116
5.3.11.4	FP File Transfer Progress.....	117
5.3.11.5	FT File Transfer.....	117
5.3.11.6	ID Identification.....	117
5.3.11.7	ST Status.....	117
5.3.11.8	Examples of using file transfer.....	118
5.3.12	Network topology attributes.....	118
5.3.12.1	LE Level Enumeration.....	118
5.3.12.2	LP Loop State.....	118
5.3.12.3	LV Level Value.....	119
5.3.12.4	LX Level Text.....	119
5.3.12.5	ND Network Topology Data.....	119
5.3.12.6	NF Network Feeds.....	120
5.3.13	Event history attributes.....	120
5.3.13.1	CA Changed Attribute	120
5.3.13.2	ED Event Daylight Saving.....	120
5.3.13.3	EM Event Time Milliseconds.....	121
5.3.13.4	EQ Qualified Event Time.....	121
5.3.13.5	ET Event Time.....	121
5.3.13.6	EX Event Comment Text.....	121
5.3.13.7	HD History Logging Daylight Saving.....	121
5.3.13.8	HM History Logging Time Milliseconds.....	122
5.3.13.9	HQ Qualified History Logging Time.....	122
5.3.13.10	HT History Logging Time.....	122
5.3.13.11	MX Message Text.....	122
5.3.13.12	PV Previous Value.....	122
5.3.13.13	US User Name.....	123
5.4	Defining process objects.....	123
5.4.1	Examples.....	123
5.5	Process object group attributes.....	124
5.5.1	CC Control Supervision Configuration.....	124
5.5.2	CD Configuration Data	125
5.5.3	GA Group Alarm.....	125

5.5.4	GB	Group Blockings.....	125
5.5.5	GC	Group Comment.....	126
5.5.6	GS	Group Alarm State.....	126
5.5.7	GT	Group Type.....	126
5.5.8	LF	Logical Format.....	127
5.5.9	LN	Logical Name.....	127
5.5.10	PV	Process Views.....	127
5.5.11	ZT	Modification Time.....	127
Section 6		Event handling objects.....	129
6.1		About this section.....	129
6.2		General.....	129
6.2.1		Different types of event handling objects.....	129
6.2.2		Use.....	129
6.2.3		Common functionality.....	130
6.2.4		AEC functionality.....	131
6.2.5		Storage.....	131
6.2.6		Event handling object notation.....	131
6.3		Common event handling object attributes.....	132
6.3.1	CX	Comment Text	132
6.3.2	HT	Event Handling Type.....	132
6.3.3	LN	Logical Name.....	132
6.3.4	MT	Message Texts.....	132
6.3.5	MX	Translated Message Texts.....	133
6.3.6	ST	State Texts.....	134
6.3.7	SX	Translated State Texts.....	134
6.3.8	VC	Value Count.....	134
6.3.9	VF	Value Formula.....	135
6.3.10	VL	Value Low.....	136
6.3.11	ZT	Modification Time.....	136
6.4		OPC Alarms & Events server attributes.....	136
6.4.1	OC	OPC Condition Event.....	136
6.4.2	OG	OPC Event Generation.....	137
6.4.3	OS	OPC Simple Event.....	138
6.5		OPC Alarms & Events client attributes.....	138
6.5.1	CD	Event Category Description.....	138
6.5.2	CI	Event Category ID.....	138
6.5.3	CN	Condition Name.....	139
6.5.4	EM	Event Messages.....	139
6.5.5	ET	Event Type.....	139
6.5.6	ND	Node.....	140
6.5.7	SN	Subcondition Names.....	140
6.6		Defining event handling objects using SCIL.....	140
6.7		Predefined event handling objects.....	140
6.7.1		OV related predefined event handling objects.....	141
6.7.2		Predefined event handling objects for non-OV events.....	142

6.8	Application default event handling objects.....	143
Section 7	Scales.....	145
7.1	About this section.....	145
7.2	General.....	145
7.2.1	Use.....	145
7.2.2	Function.....	145
7.2.3	Storage.....	145
7.2.4	Scale object notation.....	146
7.3	Scale attributes.....	146
7.3.1	LN Logical Name.....	146
7.3.2	SA Scaling Algorithm.....	146
7.3.3	SC Scaling Constants.....	146
7.3.4	ZT Modification Time.....	147
7.4	Defining scale objects using SCIL.....	147
7.4.1	Required attributes.....	147
Section 8	Data objects.....	149
8.1	General.....	149
8.1.1	Use.....	149
8.1.2	Function.....	149
8.1.3	Variables.....	150
8.1.4	Accessing registered data with SCIL.....	151
8.1.5	Execution queues.....	151
8.1.6	Storage.....	152
8.1.7	Data object notation.....	153
8.2	Data object attributes.....	153
8.2.1	Basic attributes.....	153
8.2.1.1	CM Comment.....	153
8.2.1.2	FI Free Integer.....	154
8.2.1.3	FX Free Text.....	154
8.2.1.4	IU In Use.....	154
8.2.1.5	LN Logical Name.....	154
8.2.1.6	ON OPC Item Name.....	154
8.2.1.7	ZT Modification Time.....	155
8.2.2	Logging attributes.....	155
8.2.2.1	HR History Registrations.....	155
8.2.2.2	IN Instruction.....	155
8.2.2.3	LF Logging Function.....	156
8.2.2.4	PS Pulse Scale.....	157
8.2.2.5	SR Source.....	157
8.2.2.6	TS Time Stamp.....	157
8.2.2.7	VL Value Length.....	158
8.2.2.8	VT Value Type.....	158
8.2.3	Historian logging.....	158
8.2.3.1	GN Logging Name.....	158
8.2.3.2	GP Logging Profile.....	159

8.2.4	Execution control.....	159
8.2.4.1	EP Execution Priority.....	159
8.2.4.2	PE Parallel Execution.....	159
8.2.4.3	PQ Parallel Queue.....	159
8.2.4.4	SE Start-up Execution.....	160
8.2.4.5	TC Time Channel.....	160
8.2.5	Storage.....	160
8.2.5.1	HN History File Number.....	160
8.2.5.2	MO Memory Only.....	160
8.2.6	Registered data.....	161
8.2.6.1	LR Latest Registration.....	161
8.2.6.2	OS Object Status.....	161
8.2.6.3	OV Object Value.....	161
8.2.6.4	QT Qualified Registration Time.....	162
8.2.6.5	RT Registration Time.....	162
8.3	Defining data objects using SCIL.....	162
Section 9	Command procedures.....	165
9.1	General.....	165
9.1.1	Use.....	165
9.1.2	Function.....	165
9.1.3	Program.....	166
9.1.4	Variables.....	166
9.1.5	Storage.....	166
9.1.6	Object notation.....	166
9.2	Command procedure attributes.....	167
9.2.1	Basic attributes.....	167
9.2.1.1	CM Comment.....	167
9.2.1.2	FI Free Integer.....	167
9.2.1.3	FX Free Text.....	167
9.2.1.4	IU In Use.....	168
9.2.1.5	LN Logical Name.....	168
9.2.1.6	ON OPC Item Name.....	168
9.2.1.7	ZT Modification Time.....	168
9.2.2	Program.....	169
9.2.2.1	CP Compiled Program.....	169
9.2.2.2	CS Compilation State.....	169
9.2.2.3	IN Instructions.....	169
9.2.3	Time and validation stamps.....	169
9.2.3.1	OS Object Status.....	169
9.2.3.2	QT Qualified Registration Time.....	170
9.2.3.3	RT Registration Time.....	170
9.2.3.4	TS Time Stamp.....	170
9.2.4	Execution control.....	171
9.2.4.1	EP Execution Priority.....	171
9.2.4.2	PE Parallel Execution.....	171

9.2.4.3	PQ	Parallel Queue.....	171
9.2.4.4	SE	Start-up Execution.....	171
9.2.4.5	TC	Time Channel.....	172
9.2.5		Storage attributes	172
9.2.5.1	HN	History File Number.....	172
9.2.5.2	MO	Memory Only.....	172
9.3		Defining command procedures with SCIL	172
Section 10		Time channels.....	173
10.1		General.....	173
10.1.1		Use.....	173
10.1.2		Function.....	173
10.1.3		Storage.....	175
10.1.4		Object notation.....	175
10.2		Time channel attributes.....	175
10.2.1		Basic attributes	175
10.2.1.1	CM	Comment.....	175
10.2.1.2	IU	In Use.....	176
10.2.1.3	LN	Logical Name.....	176
10.2.1.4	MO	Memory Only.....	176
10.2.1.5	ZT	Modification Time.....	176
10.2.2		Scheduling.....	176
10.2.2.1	CD	Condition.....	176
10.2.2.2	CP	Cycle Policy.....	177
10.2.2.3	CY	Cycle.....	177
10.2.2.4	DP	Daylight Switch Policy.....	177
10.2.2.5	SP	Start-up Execution Policy.....	178
10.2.2.6	SU	Synchronization Unit.....	179
10.2.2.7	SY	Synchronization Time.....	179
10.2.3		Parallel execution.....	180
10.2.3.1	PE	Parallel Execution.....	180
10.2.3.2	PQ	Parallel Queue.....	180
10.2.3.3	SX	Synchronized Execution.....	180
10.2.4		Time tagging.....	181
10.2.4.1	QB	Qualified Begin Time.....	181
10.2.4.2	QE	Qualified End Time.....	182
10.2.4.3	QS	Qualified Synchronization Time.....	182
10.2.4.4	QT	Qualified Registration Time.....	182
10.2.4.5	RB	Registered Begin Time.....	182
10.2.4.6	RE	Registered End Time.....	183
10.2.4.7	RS	Registered Synchronization Time.....	183
10.2.4.8	RT	Registration Time.....	183
10.3		Defining time channels with SCIL.....	183
Section 11		Event channels.....	185
11.1		General.....	185
11.1.1		Use.....	185

11.1.2	Function.....	185
11.1.3	Process event channels.....	186
11.1.4	Predefined event channels.....	186
11.1.5	Variables.....	186
11.1.6	Storage.....	187
11.1.7	Object notation.....	187
11.2	Event channel attributes.....	187
11.2.1	CM Comment.....	187
11.2.2	LN Logical Name.....	187
11.2.3	ON Object Name.....	188
11.2.4	OT Object Type.....	188
11.2.5	SN Secondary Object Names.....	188
11.2.6	ST Secondary Object Types.....	188
11.2.7	ZT Modification Time.....	189
11.3	Predefined event channels.....	189
11.3.1	Predefined application event channels.....	189
11.3.1.1	Application start-up and shutdown event channels.....	189
11.3.1.2	Alarm event channel.....	190
11.3.1.3	Alarm acknowledgement event channel.....	190
11.3.1.4	Blocking event channel.....	190
11.3.1.5	Monitor event channel.....	191
11.3.1.6	Undefined process object.....	191
11.3.1.7	Undefined OPC event source.....	191
11.3.1.8	Invalid OPC item.....	191
11.3.1.9	Mirroring configuration error.....	192
11.3.1.10	Host address missing.....	192
11.3.1.11	AEP_EVENT.....	193
11.3.1.12	HISTORIAN_EVENT.....	193
11.3.1.13	UAL_EVENT.....	194
11.3.1.14	AOR_EVENT.....	194
11.3.1.15	Generic application event channel.....	195
11.3.1.16	Unit events.....	195
11.3.1.17	APL-APL diagnostic events.....	196
11.3.1.18	Host events.....	196
11.3.1.19	Image events.....	197
11.3.2	Predefined system event channels.....	197
11.3.2.1	IP_EVENT.....	197
11.3.2.2	Generic system event channel.....	198
11.3.2.3	Printer events.....	198
11.3.2.4	Node events.....	199
11.3.2.5	External clock events.....	199
11.3.2.6	Application state supervision events.....	199
11.3.2.7	Memory pool supervision events.....	200
11.3.2.8	Application queue supervision events.....	200
11.3.2.9	License events.....	201
11.3.3	Operating system event channel.....	202

Section 12	Logging profile objects.....	205
12.1	General.....	205
12.1.1	Use.....	205
12.1.2	Functionality.....	205
12.1.3	Different types of logging profile objects.....	205
12.1.4	Storage.....	206
12.1.5	Object notation.....	206
12.2	Logging profile attributes.....	207
12.2.1	Common attributes.....	207
12.2.1.1	CM Comment.....	207
12.2.1.2	IU In Use.....	207
12.2.1.3	LN Logical Name.....	207
12.2.1.4	PT Profile Type.....	207
12.2.1.5	ZT Modification Time.....	207
12.2.2	Object profile attributes.....	208
12.2.2.1	CA Compression Accuracy.....	208
12.2.2.2	DI Discreteness.....	208
12.2.2.3	ST Storage.....	208
12.2.3	Database profile attributes.....	209
12.2.3.1	DA Database Address.....	209
12.2.3.2	DC Diagnostic Counters.....	209
12.2.3.3	DD Description Pattern D.....	210
12.2.3.4	DP Description Pattern P.....	210
12.2.3.5	ED Equipment Path Pattern D.....	210
12.2.3.6	EP Equipment Path Pattern P.....	210
12.2.3.7	ND Name Pattern D.....	211
12.2.3.8	NP Name Pattern P.....	211
12.2.3.9	OS Object Status.....	211
12.2.3.10	PW Password.....	212
12.2.3.11	US User Name.....	212
12.2.4	History collection profile attributes.....	212
12.2.4.1	HC History Collection Templates.....	212
Section 13	Event objects.....	213
13.1	Use.....	213
13.2	Function.....	213
13.3	Event object execution	214
13.4	Storage.....	214
13.5	Event object notation.....	214
13.6	Example.....	215
Section 14	Free type objects.....	217
14.1	About this section.....	217
14.2	General	217
14.2.1	Use.....	217
14.2.2	Attributes.....	217

14.2.3	Object definition.....	217
14.2.4	Storage.....	218
14.2.5	Object notation.....	218
14.3	Type defining attributes.....	219
14.3.1	LN Logical Name.....	219
14.3.2	PT Process Object Type.....	219
14.3.3	OV OV Attribute Name.....	219
14.3.4	CX Comment Text.....	219
14.3.5	NA Number of Attributes.....	220
14.3.6	ZT Modification Time.....	220
14.4	Attributes for defining attributes.....	220
14.4.1	AN Attribute Name.....	220
14.4.2	AI Attribute Indexing.....	220
14.4.3	AT Attribute Value Type.....	221
14.4.4	AL Attribute Length.....	221
14.4.5	AP Attribute Printout.....	221
14.4.6	AA Attribute Action.....	222
14.4.7	AH Attribute History.....	222
14.4.8	AE Attribute Event.....	222
14.4.9	AD Attribute on Disk.....	222
14.4.10	AS Attribute Snapshot.....	222
14.4.11	AX Attribute Comment Text.....	223
14.4.12	AO Attribute Offset.....	223
14.5	Defining free type objects.....	223
14.5.1	Examples.....	223

Section 15 Variable objects.....225

15.1	Use.....	225
15.2	Definition.....	225
15.3	Function.....	225
15.4	Attributes.....	225
15.5	Variable object notation.....	226
15.6	Storage.....	226
15.7	Examples.....	226

Section 16 Using object definition tools.....229

16.1	Application Object Navigator.....	229
16.1.1	Entering and exiting object navigator.....	229
16.1.2	Object navigator composition.....	229
16.1.3	Status bar.....	230
16.1.4	Accessible applications.....	233
16.1.5	External applications.....	234
16.1.6	Proxy applications.....	235
16.1.7	Representation options for objects.....	236
16.1.8	Page length.....	238
16.1.9	Auto adjust table columns.....	238
16.1.10	Refresh functionality.....	238

16.1.11	Filtering names.....	239
16.1.12	Filtering objects.....	239
16.1.13	Defining a Filter.....	239
16.1.14	Editing attribute values of objects.....	241
16.1.15	Viewing or editing objects.....	242
16.2	Navigation.....	243
16.2.1	Overview.....	243
16.2.2	Navigation By Unit.....	243
16.2.3	Navigation By Unit and Item Name.....	244
16.2.4	Navigation By Node and Item Name.....	244
16.2.5	Navigation By Object Identifier.....	245
16.2.6	Navigation By OPC Name.....	246
16.2.7	Navigation By Event Source.....	247
16.3	Creating and editing objects.....	247
16.3.1	Creating new application objects.....	247
16.3.2	Creating new OPC Event objects.....	249
16.3.3	Creating new event handling objects.....	250
16.3.4	Creating new data objects.....	251
16.3.5	Creating new logging profile objects.....	251
16.3.6	Event recording objects.....	252
16.3.7	Defining an application object.....	252
16.3.8	Defining a process object group.....	252
16.3.9	Renaming application objects.....	253
16.3.10	Copying application objects.....	253
16.3.11	Copying process objects index.....	255
16.3.12	Copying bay level process objects	258
16.3.13	Editing process object identifiers by levels.....	259
16.3.14	Moving objects.....	260
16.3.15	Deleting application objects	260
16.3.16	Installing standard functions.....	261
16.3.17	Documenting application objects.....	261
16.3.18	Exporting and importing application objects.....	261
16.3.19	Searching objects and files.....	261
16.3.20	File transfer.....	261
16.4	General principles for using object definition tools.....	263
16.4.1	Definition tool title.....	263
16.4.2	Pages.....	264
16.4.3	User interaction	264
16.4.4	Storing the object and exiting the tool.....	264
16.4.5	Connecting and disconnecting objects.....	265
16.5	Application self-diagnostics.....	269
Section 17	Process Object Definition Tool.....	271
17.1	About this section.....	271
17.2	General.....	271
17.3	Common area.....	271

17.3.1	General.....	271
17.3.2	Identification.....	272
17.3.3	Operation state.....	272
17.3.4	Process signal type.....	272
17.3.5	Other information.....	273
17.4	Configurable attributes.....	273
17.4.1	Overview.....	273
17.4.2	Addresses.....	274
17.4.3	Scaling.....	274
17.4.4	Limit values.....	275
17.4.5	Alarms.....	275
17.4.6	Post-processing.....	276
17.4.7	Events.....	277
17.4.8	History.....	277
17.4.9	Printouts.....	278
17.4.10	Blocking.....	278
17.4.11	Topology.....	279
17.4.12	Miscellaneous.....	279
17.5	Dynamic attributes.....	279
17.5.1	Object state.....	279
17.5.2	File transfer.....	281
17.5.3	Value history.....	282
17.5.4	OPC event.....	282
17.5.5	Alarm.....	283
17.5.6	Counters.....	283
17.6	All attributes.....	284
17.7	Object Identifier editor.....	284
Section 18	Event Handling Object Definition Tool.....	287
18.1	About this section.....	287
18.2	General.....	287
18.3	Common area.....	287
18.3.1	Object state calculation area.....	288
18.3.2	OPC A&E event definition.....	288
18.4	State and message texts.....	289
18.4.1	Overview.....	289
18.4.2	State texts.....	289
18.4.3	Message texts.....	289
18.5	Event messages and subcondition names.....	290
18.5.1	Overview.....	290
18.5.2	Event messages.....	290
18.5.3	Subcondition names.....	290
18.6	OPC A&E server.....	291
18.6.1	Overview.....	291
18.7	Listed values.....	292
Section 19	Scale Object Definition Tool.....	295

19.1	About this section.....	295
19.2	General.....	295
19.3	Common area.....	295
19.4	Linear scaling.....	295
19.5	Stepwise linear scaling.....	296
Section 20	Data Object Definition Tool.....	297
20.1	About this section.....	297
20.2	Overview.....	297
20.3	Common area.....	298
20.4	Data registration.....	298
20.5	Data.....	299
20.6	Execution control.....	301
20.7	Storage.....	301
20.8	External logging.....	302
20.9	All attributes.....	302
Section 21	Command Procedure Definition Tool.....	305
21.1	Overview.....	305
21.2	Common area.....	305
21.3	Procedure page.....	305
21.4	Execution control page.....	306
21.5	Storage page.....	307
21.6	All attributes.....	307
Section 22	Time Channel Definition Tool.....	309
22.1	About this section.....	309
22.2	Overview.....	309
22.3	Common area.....	309
22.4	Execution.....	309
22.5	Initialization.....	310
22.6	Execution control.....	311
22.7	Connected objects	312
22.8	All attributes.....	313
Section 23	Event Channel Definition Tool.....	315
23.1	About this section.....	315
23.2	Overview.....	315
23.3	Common area.....	315
23.4	Activated objects.....	315
23.5	Connected objects.....	316
23.6	Attribute list.....	317
Section 24	Logging Profile Object Definition Tool.....	319
24.1	About this section.....	319
24.2	General.....	319
24.3	Definition tool of the logging profile object of type OBJECT.....	319
24.4	Definition tool for logging profile object of type DATABASE.....	320

24.5	Definition tool for logging profile object of type HISTORY.....	321
Section 25	Free Type Object Definition Tools.....	323
25.1	General.....	323
25.2	Free Type Process Object Tool.....	323
25.2.1	Accessing the tool.....	323
25.2.2	Fields.....	324
25.2.3	Using the tool.....	325
25.2.4	Storing settings and exiting the Tool.....	325
25.3	Free Type Object Tool.....	325
25.3.1	Accessing the tool.....	325
25.3.2	Using the tool.....	325
25.3.3	Attribute definition.....	325
25.3.4	Attribute tree.....	326
25.3.5	Storing settings and exiting the tool.....	327
Index.....		329

Section 1 Copyrights

The information in this document is subject to change without notice and should not be construed as a commitment by Hitachi Power Grids. Hitachi Power Grids assumes no responsibility for any errors that may appear in this document.

In no event shall Hitachi Power Grids be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall Hitachi Power Grids be liable for incidental or consequential damages arising from the use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without written permission from Hitachi Power Grids, and the contents thereof must not be imparted to a third party nor used for any unauthorized purpose.

The software or hardware described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

© 2021 Hitachi Power Grids. All rights reserved.

Trademarks

ABB is a registered trademark of ABB Asea Brown Boveri Ltd. Manufactured by/for a Hitachi Power Grids company. All other brand or product names mentioned in this document may be trademarks or registered trademarks of their respective holders.

Guarantee

Please inquire about the terms of guarantee from your nearest Hitachi Power Grids representative.

Third Party Copyright Notices

List of Third Party Copyright notices are documented in "3rd party licenses.txt" and other locations mentioned in the file in SYS600 and DMS600 installation packages.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<https://www.openssl.org/>). This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Section 2 About this manual

2.1 Use of symbols

This publication includes warning, caution and information symbols where appropriate to point out safety-related or other important information. It also includes tips to point out useful hints to the reader. The corresponding symbols should be interpreted as follows:



Warning icon indicates the presence of a hazard which could result in personal injury.



Caution icon indicates important information or a warning related to the concept discussed in the text. It might indicate the presence of a hazard, which could result in corruption of software or damage to equipment/property.



Information icon alerts the reader to relevant factors and conditions.



Tip icon indicates advice on, for example, how to design a project or how to use a certain function.

Although warning hazards are related to personal injury, and caution hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all warnings and caution notices.

2.2 Intended audience

This manual is intended for installation personnel, administrators and skilled operators to support installation of the software.

2.3 Related documents

Name of the manual	Document ID
SYS600 10.2 Application Design	1MRK 511 466-UEN
SYS600 10.2 Installation and Administration	1MRK 511 496-UEN
SYS600 10.2 Programming Language SCIL	1MRK 511 479-UEN
SYS600 10.2 System Objects	1MRK 511 482-UEN
SYS600 10.2 System Configuration	1MRK 511 481-UEN
SYS600 10.2 OPC Server	1MRK 511 476-UEN
SYS600 10.2 Historian Configuration and Administration	1MRK 511 472-UEN
SYS600 10.2 Historian Operation	1MRK 511 474-UEN
SYS600 10.2 Historian Monitor Configuration	1MRK 511 473-UEN

2.4 Document conventions

The following conventions are used for the presentation of material:

- The words in names of screen elements (for example, the title in the title bar of a dialog, the label for a field of a dialog box) are initially capitalized.
- Capital letters are used for file names.
- Capital letters are used for the name of a keyboard key if it is labeled on the keyboard. For example, press the CTRL key. Although the Enter and Shift keys are not labeled, they are written in capital letters, for example, press ENTER.
- Lowercase letters are used for the name of a keyboard key that is not labeled on the keyboard. For example, the space bar, comma key and so on.
- Press CTRL+C indicates that the user must hold down the CTRL key while pressing the C key (in this case, to copy a selected object).
- Press ALT E C indicates that the user presses and releases each key in sequence (in this case, to copy a selected object).
- The names of push and toggle buttons are boldfaced. For example, click **OK**.
- The names of menus and menu items are boldfaced. For example, the **File** menu.
 - The following convention is used for menu operations: **Menu Name/Menu Item/Cascaded Menu Item**. For example: select **File/Open/New Project**.
 - The **Start** menu name always refers to the **Start** menu on the Windows Task Bar.
- System prompts/messages and user responses/input are shown in the Courier font. For example, if the user enters a value that is out of range, the following message is displayed: Entered value is not valid.
- The user may be told to enter the string MIF349 in a field. The string is shown as follows in the procedure: MIF349
- Variables are shown using lowercase letters: sequence name

2.5 Document revisions

Revision	Version number	Date	History
A	10.2	31.03.2021	New document for SYS600 10.2

Section 3 Introduction

3.1 About this section

This section introduces the MicroSCADA X SYS600 system and describes the role of application objects.

3.2 SYS600

SYS600 is a computer-based, programmable, and scalable network control and substation automation system. It is mainly used within the electric power process but can also be used for the supervision and control of heat and water distribution, industrial processes, water purification, traffic, etc.

The system servers of SYS600 are composed of the kernel, a number of facility programs, engineering and system handling tools, configuration software and application software. The SYS600 kernel software is independent of the application area and the extent of use. It is the same in all systems, as are also most of the engineering and system handling tools. A system server can host one or more application software packages, called applications. The application specifies the functions of the SYS600 system to suit a specific process. The application takes the user's needs into account regarding the level of information, user interface, control operations, and so on.

3.3 Applications

Each application has a certain supervisory control task, for example the control of electricity distribution or heat distribution. An application may control its own process and have its own connections to the process equipment, or it may share the equipment with other applications. Each application has its own data (databases) and displays. Different applications can communicate with each other, whether they are situated in the same system server or in separate ones. In simple terms, an application is composed of a set of objects that communicate with each other, with the user and with the process equipment, see "A simplified scheme of a SYS600 application" in [Figure 1](#).

The application objects specify control functions, calculations, data storage, process control etc., and they are composed of process data (process objects), report data (data objects), control programs (command procedures) and activation mechanisms (event channels, time channels, logging profiles, events and event handling objects).

The application objects are programmed and controlled using the SCIL language, which is an application language specifically developed for SYS600 and the control & supervision domain.

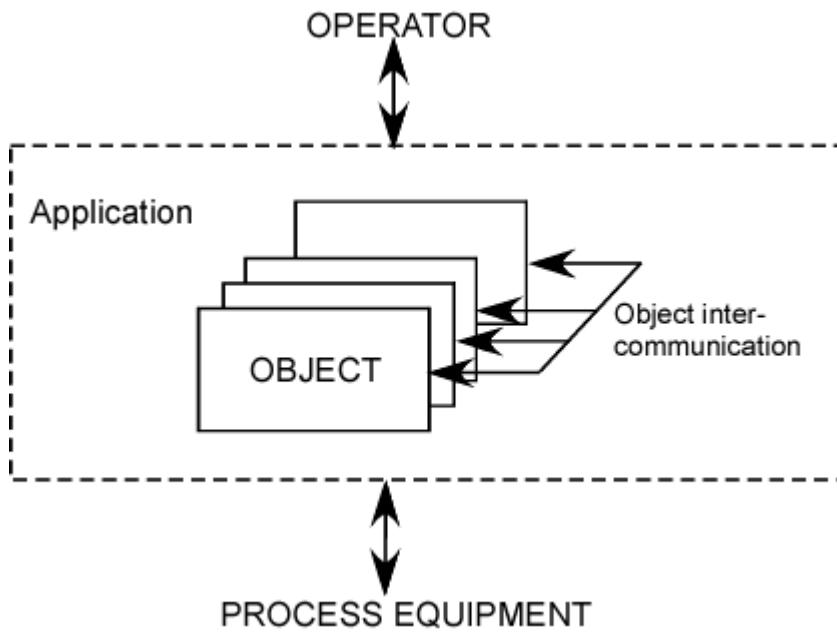


Figure 1: A simplified scheme of a SYS600 application

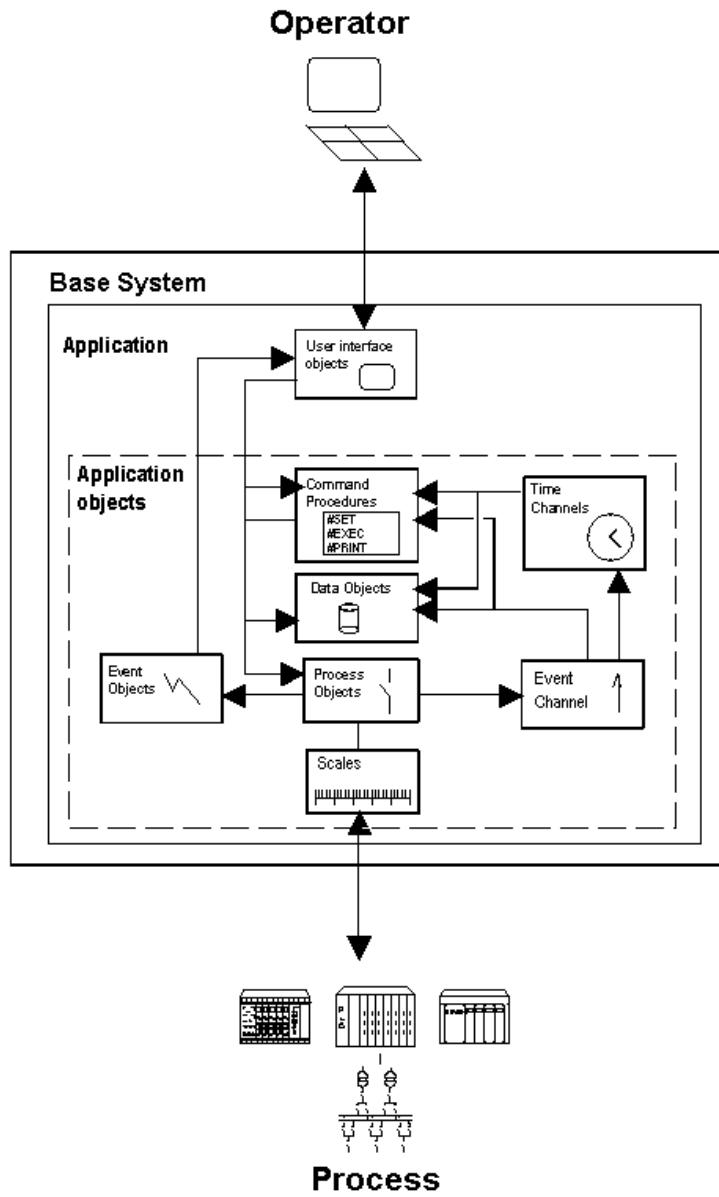
3.4 Application objects

Application objects are programmable units that perform various tasks, such as real time process supervision, control procedures, data registration and storage, calculations, automatic time and event activation, etc. There are eleven types of application objects, each performing a particular task:

1. Process objects (P). Process objects are images of connected process signals. These objects store and supervise the real time state of a process.
2. Event handling objects (H). Event handling objects define the texts related to the states of process objects and transitions between the states (events).
3. Scales (X). Scales are algorithms for scaling the data transferred from the stations to the real values of the measured entity.
4. Data objects (D). Data objects register and store sampled or calculated data.
5. Command procedures (C). These objects are SCIL programs, which can be executed automatically or manually.
6. Time channels (T). These objects control the automatic time-based data registrations and program executions.
7. Event channels (A). These objects control automatic event-based data registration and program execution.
8. Logging profiles (G). These objects define the connection between SYS600 application database and Historian time series database.
9. Event objects (E). These objects activate automatic event-controlled program execution (for example updating) in user interface objects.
10. Free Type objects (F). Free Type objects define user-defined process object types.
11. Variable objects (V). Variable objects are temporary lists of attributes and attribute values.

The first seven types are illustrated in [Figure 2](#).

The capital letter after each type in the list above is an identification mark for the object type when used in SCIL. All types of application objects, except variable objects, are global and accessible using SCIL throughout the entire SYS600 network.



Appo_interconnections.gif

Figure 2: An illustration of the application objects and their interconnections

3.5 Attributes

Information and data associated with objects, their values, functions, properties and activities, are stored in attributes. Normally, an object has many different attributes, and thus it can contain several types of data. Different object types have different sets of attributes.

The attributes not only contain the dynamic data of the objects, but also define the objects and their functions. Static (defining) attributes include object identities, addresses, activation criteria, activity states, connections to other objects, alarm handling specifications, SCIL programs and expressions. Examples of dynamic attributes are the object values, historical data, status codes and time tags. [Figure 3](#) shows an imaginary object (data object) and its static and dynamic attributes.

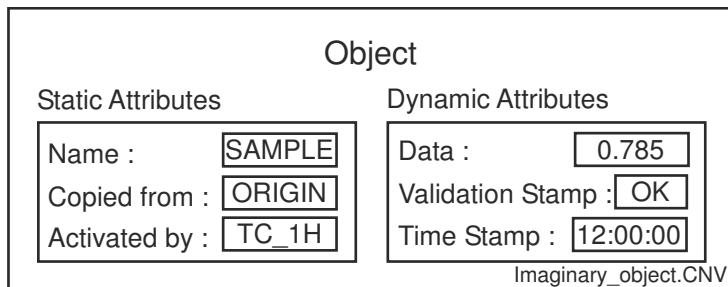


Figure 3: An imaginary object (data object) and a few of its static and dynamic attributes

Attribute values can be used in SCIL expressions and programs, for example in calculations, displays, conditional statements, etc. The values can be changed automatically by the process or system, or manually using SCIL. All dynamic and static attributes can be accessed using SCIL, though all of them cannot be set using it.

As a rule, attributes are the only way to store, access, use and modify information in objects and to operate through them. Attributes are therefore the most essential part of objects.

3.6 Defining application objects

Creation and definition of application objects is a part of application engineering, that is the composition of an application software package. The object definitions may also be modified in a running application. Objects can be created and modified as follows:

- Using SCIL commands. This is the basic method and the other methods are based on this one, because the tools are built with SCIL.
- Using type specific definition tools. In the object definition tools, objects are defined by filling in data and making appropriate choices.
- Using the standard application library LIB 500. Creation of objects using the standard application library is mostly invisible to the user.
- Using Connectivity Packages or SCL files.

Event objects are event-based activation signals that trigger execution of event methods of Visual SCIL objects and event handlers (ON blocks) of pictures. They are generated either by process objects or by SCIL. These objects have no attributes and no other definitions beside activation criteria and the object name when activated by SCIL. The picture program parts activated by the event objects are stored within the pictures in the picture database and can be activated only when the picture is visible on the screen.

Variable objects are always created with SCIL. The variable objects are used as variables and as temporary storage for data.

3.7 Databases

Most application object types are stored in **databases**. A database is a set of related data stored in a structured form.

The process objects, event handling objects, scales and free type objects are stored in the **Process Database**. The data objects, command procedures, time channels, event channels and logging profiles are stored in a database named **Report Database**, and these objects are called report objects with a common name.

Variable objects are stored in the same way as variables (see the Programming Language SCIL manual). Event objects are not stored.

Each application may only contain one process database and one report database.

Section 4 Object handling

This section presents the principles on how to define application objects and how to use the objects in SCIL. It is divided into the following three sections:

- [Section 4.1](#) [Defining application objects 4.1](#): the possibilities for defining objects, the principles for using the object definition tools, the principles for defining objects with SCIL.
- [Section 4.2](#) [Using application objects in SCIL 4.2](#): the object notation and how to use object notations in SCIL, accessing attributes.
- [Section 4.3](#) [Some SCIL commands 4.3](#): for application object handling.

4.1 Defining application objects

As mentioned in [Section 3](#), application objects can be created (defined) and modified in several ways:

- Using application object definition tools.
- Using standard application libraries, such as LIB 500.
- Using Connectivity Packages or SCL files.
- By writing SCIL programs.

4.1.1 Using definition tools

By using application object definition tools, the engineer defines the objects, one by one, by entering attributes in fields or by making selections from various options.

The application object definition tools are accessed from the Tool Manager by double-clicking the Object Navigator. In the Object Navigator the user can view object lists, access the definition of selected objects, add new objects, copy objects within the same application or from one application to another, delete objects, etc.

The application object definition tools are described at the end of this manual. The Object Navigator and the principles for using the object definition tools are discussed in [Section 16](#).

Although objects have been created with SCIL or with the LIB 500 tools, they can be viewed and edited by using the object definition tools.

4.1.2 Using LIB 500

LIB 500 is a set of standard application libraries for fast standardized application engineering. The LIB 500 libraries contain various types of application components. The application engineer uses these components to compose a complete application. The composition is done using the picture based library tools. The library tools create required application objects collectively. The programmer only gives some basic information.

4.1.3 Using Connectivity Packages or SCL files

Connectivity Packages for IED's or IEC 61850 SCL files can be used to create process objects structured according to Power Process (SA-LIB) standard functions such as switching devices,

measurements, etc. The SCL files contain the description of an IED, a part of a substation or even a complete substation. The information in this file is generated by the IED configuration tool(s). SYS600 contains a SCL importer tool that can read the SCL file and create the process database accordingly. The Connectivity Package engineering method is described in the Application Design manual.

4.1.4 Defining objects with SCIL

New application objects can be created with the SCIL command #CREATE. Objects can be modified with the #MODIFY command and objects can be deleted with the #DELETE command. See the brief description of SCIL commands in [Section 3](#), or the complete description in the Programming Language SCIL manual.

Application objects are always created, modified and deleted with SCIL, since the object engineering tools are built with SCIL.

4.2 Using application objects in SCIL

4.2.1 General

In SCIL the application objects are used mainly through their attributes. Object data can be included in SCIL programs and expressions via attributes. Object data can, for instance, be shown in windows, it can form the basis for control operations or be used in the definition of other objects, etc. Objects are identified by an **object notation** and attributes of an object by an **object attribute notation**, see below.

All types of objects can be used in SCIL. However, event objects cannot be included in expressions as they have neither data nor attributes.

4.2.2 Object notation

Object notation has the following format:

name:{application}type{index}

where

'name'	is the object name
'application'	is the logical application number
'type'	is the object type
'index'	is an index number or an index range

The components of the notation are explained below. The ones within braces are not always required. Indexing is applicable only for process objects, all the other objects are unindexed. See the examples below in [Section 4.2.9](#).

4.2.3 Object attribute notation

Object attribute notation has the following format:

name:{application}type{attribute}{index}

where

'name'	is the object name
'application'	is the logical application number
'type'	is the object type
'attribute'	is the attribute name
'index'	is an index number or an index range

The components of the notation are explained below. The ones within braces are not always required. The attribute name may be omitted if the object type has a default attribute. For process objects of a predefined type, indexing is used to select the object. For other types of objects, indexing is used to access elements of a vector valued attribute. See the examples below in [Section 4.2.9](#).

4.2.4 Name

The **name** of an object may be up to 63 characters long. Characters allowed are the letters A-Z, all digits, underscore (_) and period (.). The object name must begin with a letter, a digit or an underscore.

Object names can be freely chosen. Within an application, the object name must be unique for a specific object type, but objects of different types may have the same name.

Examples of correct object names:

RELAY_123
1.RELAY
BREAKER_92
PICCADILLY_LINE_BREAKER

Examples of incorrect object names:

RELÄ	Contains an extended character (Scandinavian letter)
BREAKER 1	Contains a space

Process objects defined in LIB 500 must obey tighter rules: a maximum of nine characters is allowed for relay picture functions and ten characters for all the other types of picture functions.

4.2.5 Application

Application is the logical number of the application where the object is stored. It is the application number as known to the present application (according to the application mapping, the APL:BAP attribute, see the System Objects manual). The number can be omitted when the object is in the current application (the normal case).

Including an application number (other than the current one) in an object notation brings about a data transfer between two applications, within the same or different base systems. A prerequisite is that the applications recognise each other through application mapping.

4.2.6 Type

The object **type** is indicated with a letter in accordance with the following:

P	Process objects
H	Event handling objects
X	Scale objects
D	Data objects
C	Command procedures
T	Time channels
A	Event channels
G	Logging profiles
E	Event objects
V	Variable objects
F	Free type objects

4.2.7 Attribute

An **attribute** represents the value or feature to be read or written with the object notation. It is generally named by a predefined **attribute name**, which is a combination of two letters, A ... Z. Variable objects can have freely chosen attribute names of any length up to 63 characters.

Reading an attribute means that the attribute value is used in an expression. **Writing an attribute** means that the attribute value is changed or updated with the #SET or #MODIFY commands. See the examples in [Section 4.2.9](#).

The attribute of an object notation determines value and data type of the entire notation (see the Programming Language SCIL manual). An object notation without an attribute may still refer to a special attribute, the default attribute (mentioned in the subsequent object descriptions, normally the object value).

Object notations are used without an attribute together with some commands ([Section 4.3](#)). In these cases, they refer to the entire object. Event objects can only be used without an attribute.

4.2.8 Index

Indices are used to differentiate attribute values with equal object notations in all other respects. Such attribute values are handled as vectors, where the elements are accessed using indices.

As a rule, indices refer to the elements of an attribute of vector type. The actual attribute determines the data type of the elements. Predefined process object types are an exception. For these objects, the indices refer to the individual objects in a group, not to the attributes of a vector type. However, for a certain attribute, the values are handled as elements in a vector.

In SCIL, an index or index range is marked in any of the following ways:

- With an integer number, either a positive decimal number or an octal number. An index of a variable object where the attribute is not composed of two letters must be embraced by brackets. In all other cases, no brackets are needed.
- With an integer type expression. The expression must be embraced by brackets.
- With an interval (i .. j), where "i" is the first index number and "j" the last. Two points surrounded by brackets (..) are interpreted as all the indices of the actual object notation. Interval (i ..) indicates all indices larger than or equal to "i", and (.. j) all indices less than or equal to "j".

No space is allowed between the index and the rest of the object notation.

4.2.9 Using object notations

Object and object attribute notations are used in SCIL commands and expressions.

In SCIL commands, object notation is used to identify the object and object attribute notation to identify both an object and one of its attributes. For examples in using object notation, see [Section 4.3](#).

When object attribute notation is used as an operand in an expression, the value of the attribute replaces the entire notation. Examples:

```
!SHOW WINDOW OBJ:POV2
```

The value of the OV attribute of the process object OBJ with index 2 is read from the process database and displayed in the window WINDOW.

```
V = DATA:DRT + 60
```

A variable is assigned the value of the latest registration time of a data object added by 60 seconds.

```
#SET BREAKER:PBO3 = 1
```

The process object BREAKER is closed. If the object represents a real physical object, the command is sent to the relay, which closes the real breaker.

```
.SET ITEM._TITLE = "DAT_OBJ(3) " + DEC(DAT_OBJ:2DOV3)
```

The third registered value of the data object DAT_OBJ in application 2 is shown in the title of a Visual SCIL object.

4.2.10 Attribute access level

There are four main levels of access to the application object attributes:

- **Read-only:** The attribute can be read using the object notation. It cannot be written with the #SET or #MODIFY commands, nor can it be given a value when creating new object with #CREATE.
- **Read-only, configurable:** The attribute can be read and it can be assigned a value when a new object is created with the #CREATE command or modified with the #MODIFY command. It cannot be assigned values with the #SET command.
- **Read, conditional write:** The attribute can be read. It can be written provided that certain conditions are fulfilled.
- **No restrictions:** The attribute can be both read and written freely, in some cases provided that the object is in use (IU = 1).

These terms are used in the attribute descriptions in the subsequent sections.

4.3 Some SCIL commands

4.3.1 General

A detailed description of SCIL is found in the Programming Language SCIL manual. This section briefly describes a few SCIL commands and functions, which are important for application object handling. The commands here are given along with the arguments necessary for making a complete statement. The arguments are written in lower case letters.

4.3.2 Setting attribute values

4.3.2.1 #SET attribute {= expression}

Assigns the value of 'expression' to the attribute 'attribute', which must be an object attribute notation. The object can be of any object type, except for an event object. Concerning process objects, the command may entail control of the process.

4.3.3 Executing objects

4.3.3.1 #EXEC object {(variable_list)}

Executes an object. The 'object' can be a data object, a command procedure, an event object or an event channel. The command stores a new value in a data object, executes a command procedure, generates an event or starts an event channel. The variable list, which can be omitted, defines the variables to be used in the command procedure or in the data object expression.

4.3.4 Event object handling

4.3.4.1 #ON event_object [statement]

Defines a statement or a program sequence to be executed each time the named event is generated ([Section 12](#)). This command may be used in user interface objects (pictures and Visual SCIL objects) to immediately respond to an event.

4.3.5 Updating process database

4.3.5.1 #GET object

Gets process values from stations connected on ANSI lines (station type STA) and updates them in the process data base. The 'object' must be a process object or a communication system object with the attribute ME (see the System Objects manual).

4.3.6 Process queries

4.3.6.1 APPLICATION_OBJECT_LIST, APPLICATION_OBJECT_ATTRIBUTES, APPLICATION_ALARM_LIST, HISTORY_DATABASE_MANAGER

These SCIL functions constitute the basis for browsing through the process database, report database, alarm list and event list (history database).

4.3.7 Creating objects

4.3.7.1 #CREATE object [= expression]

Creates a new object and assigns it the attributes of the expression, which must be of list type. All types of application objects, except for event objects, can be created by using this command.

4.3.8 Modifying objects

4.3.8.1 #MODIFY object = expression

Changes the object definition according to the attributes in the list type expression. The object can be of any application object type, except for an event object. Unlike the #SET command, the #MODIFY command allows several attributes to be written simultaneously. Some attributes that cannot be changed with #SET can be changed with #MODIFY.

4.3.9 Deleting objects

4.3.9.1 #DELETE object

Deletes the object. The object can be of any application object type, except for an event object.

Section 5 Process objects

This section describes the process objects and their attributes. It is divided into five sections with the following contents:

- | | |
|-----------------------------|--|
| Section 5.1 | General: This section describes the basic features, use and functions of process objects, process object types, user defined process object types, an overview of process object attributes, the storage of process objects, etc. |
| Section 5.2 | Configurable process object attributes: This section lists and describes in detail the attributes that define the functionality of the object. |
| Section 5.3 | Dynamic process object attributes: This section lists and describes in detail the attributes that contain the dynamic real-time data of the object. |
| Section 5.4 | Defining process objects: This section lists the required attributes, default values and principles for creating process objects using SCIL, and presents examples. |
| Section 5.5 | Process object group attributes: This section lists and describes in detail the attributes that define the functionality of object groups. |

5.1 General

5.1.1 Use

Process objects are typically data images of physical process devices, such as breakers, disconnectors, switches, relays, detectors, sensors, regulators. These devices are connected to SYS600 through Remote Terminal Units (RTUs), Protective Equipment, Programmable Logics, etc., all of which will be referred to as process units or stations in the following text.

Process objects supervise the process signals registered in stations and control the signals sent from the stations to the process equipment. Generally, each input and output connection in a station is represented by a process object in the SYS600 process database. Additionally, general supervision and status information stored in a station can be represented by process objects.

There are also process objects that have no physical correspondence, nor any data correspondence in the stations. These process objects are used for process simulation, for manually updated values, system message handling, etc.

5.1.2 Function

Process objects constitute the links between the control system and the controlled process. A process object contains the process data, various stamps related to the data (for example time and validation stamps or stamps set by the stations) and the alarm state information. It also contains functional definitions, such as scale definition, automatic activation, etc., see [Figure 4](#). Both the dynamic data that reflect the real-time state of the process and the functional definitions are defined by attributes.

A process device is controlled by setting the object value of the corresponding output object with the #SET command. The order is passed out to the NET unit and to the process device via the station. Likewise, a spontaneous message from a station updates the corresponding input objects in the process database. Under certain conditions, the process objects of input type can also be updated using SCIL (with #SET, see the SS attribute). Process objects without process connection are always updated using SCIL.

Every update of a process object, whether it comes from the process or from SCIL, may cause the following effects, “post-processing”, (depending on the process object definition and the value of the update):

- Alarm activation (input objects) including alarm signals, alarm printout and registration in the alarm buffer (alarm list).
- Automatic printout.
- Event based updating of user interface objects (through event objects).
- Activation of an event channel.
- Registration in the history database (event list).
- Registration in the Historian time series database.

Each update of the object value also updates a validation stamp (the OS attribute) and a time stamp (the RT, RM and RQ attributes), and, depending on the communication protocol, possibly some other quality attributes.

If the process station sends an update message using an address that is not found in the process database, an event channel (UNDEF_PROC or UNDEF_OPC_EVENT) is activated to inform the application about the mismatch, see [Section 11](#).

5.1.3 Process object types

The type of a process object depends on the type of the corresponding input/output signal in the station. The signal corresponds to the main attribute of the process objects, the object value (the OV attribute). SYS600 supports twelve predefined process object types:

- Analog Input
- Binary Input
- Digital Input
- Double Binary Indication (input)
- Pulse Counter (input)
- Analog Output
- Binary Output
- Digital Output
- Bit Stream (input and output)
- File Transfer (input and output)
- OPC Event (input)
- Network Topology

[Table 1](#) shows the relationship between the process object types and the corresponding signal types in S.P.I.D.E.R. RTU and SPACOM stations.

5.1.4 User defined types

In addition to the predefined process object types listed above, the SYS600 application engineer can define up to 156 different user-defined types. These process object types are defined by free type objects described in [Section 14](#).

Using free type objects, the application engineer can define their own process object types for which they can select the type of object value, other attributes and type of activation. Unlike the predefined object types, whose post-processing is always connected to the object value, any attribute of the user defined object types can be specified to cause post-processing.

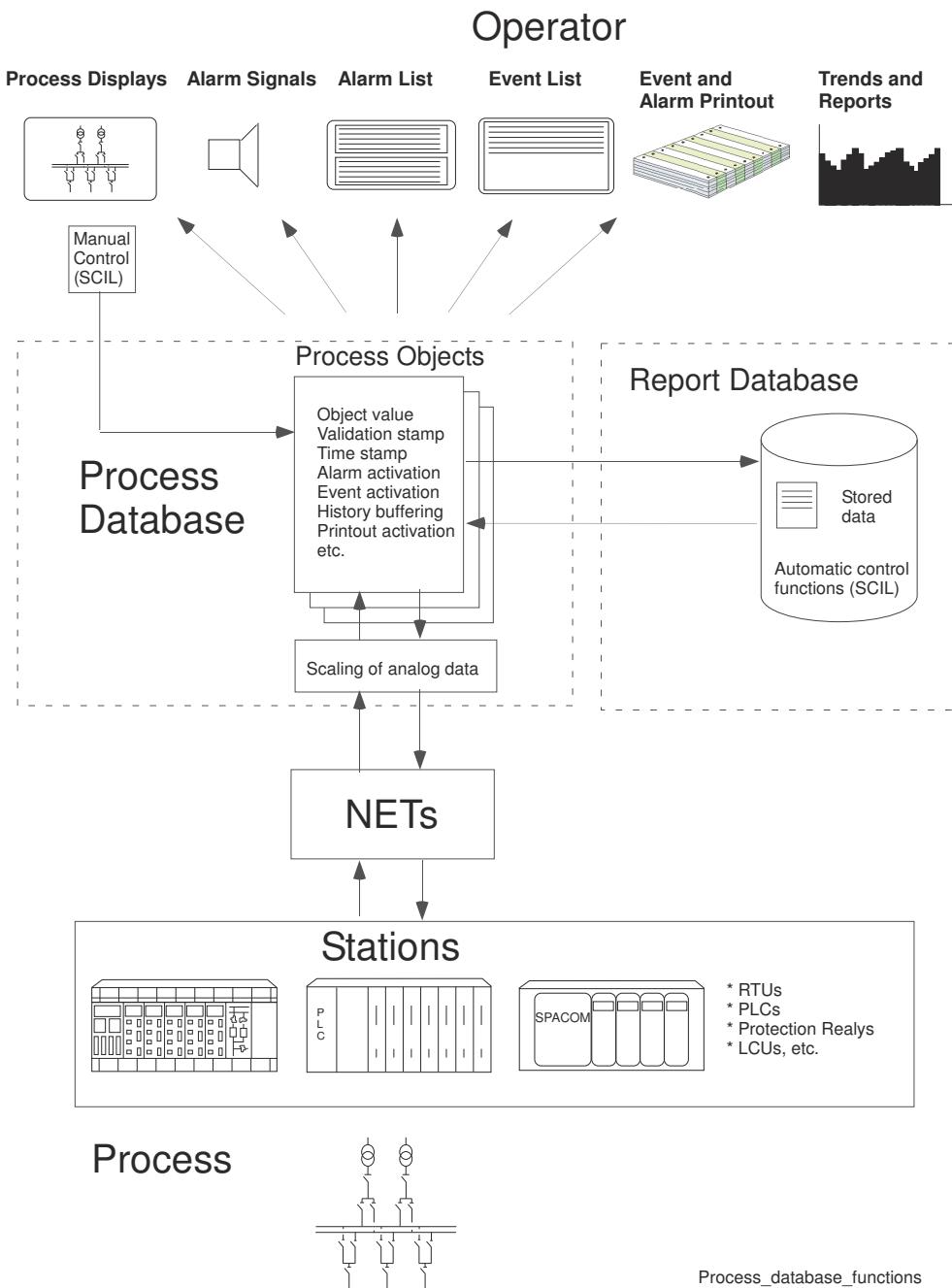


Figure 4: The functions of the process database

Table 1: The S.P.I.D.E.R. RTU and SPACOM specific data types and the corresponding process object types. The Bit Stream, File Transfer and OPC Event object types are not included as they have no correspondence within these types of stations.

Process Object Types	S.P.I.D.E.R RTU Data Types	SPACOM Data types
Binary Input, BI	Indication single Indication single event recording	Indications Binary Signals Alarms Switches
Binary Output, BO	Object command Regulation command	Select open/close of secured control Execute command Cancel selection Direct open/close commands Lower/raise commands
Digital Input, DI	Digital value	
Digital Output, DO	Digital setpoint	
Analog Input, AI	Analog value Analog event recording	Measured data (current, voltage, integrated energy values, tap changer position, etc.).
Analog Output, AO	Analog setpoint General persistent output	
Double Binary Indication, DB	Indication double Indication double event recording	Breaker and disconnector states (open + close)
Pulse Counter, PC	Pulse counter	Pulse counters

5.1.5 Attributes

[Table 2](#) lists the attributes of different process object types. This table lists both the dynamic attributes that reflect the real-time state of the process and the definition attributes that specify the functionality of the object. In the attribute descriptions in [Section 5.2](#) and in [Section 5.3](#) these two main types of attributes are kept apart. The attributes specific to OPC Event objects are omitted. They are described in [Section 5.3.10](#).

If no value is given to an attribute when an object is defined, the attribute is assigned a default value. The default values of the attributes are given in the attribute descriptions in [Section 5.2](#) and in [Section 5.3](#).

Objects of the user-defined types have all the common attributes. In addition, they have a number of user-defined attributes that can have any two-letter attribute name different from the common attribute names. The user-defined attributes are defined by the free type objects described in the [Section 14](#).

Table 2: The process object attributes of different process object types.

Object Types	All Types	Predefined types											
		BI	BO	AI	AO	DI	DO	DB	PC	BS	FT	OE	NT
Basic Attributes	LN, IX, PT, ZT												
Identification	CX, ES, IE, IL, OI, ON, OX, TX												
Object Addresses	UN, OT, TI	OA, OB, IN, IG											
Object Value	OV, SX	BI, TS	BO	AI	AO	DI	DO	DB, TS	PC	BS	FT	OE	NT

Table continues on next page

Object Types		All Types		Predefined types									
Attributes		BI	BO	AI	AO	DI	DO	DB	PC	BS	FT	OE	NT
Time and Validation Stamps	OS, RT, RM, RQ		CS		CS		CS						
Alarm State	AL, AS, AR, AV, AT, YT, AM, YM, AQ, YQ												
Operation State	IU, SS, SU												
Unit and Scale				SN, ST, DP, IR	SN, ST, DP, IR				SC, BC				
Limit Values				HI, HW, LI, LW, SZ, ZE, ZD	HO, LO								
Alarm Handling	AC, AD, PI, PD, RC	AG		WC, WR				LA, NV					LA
Min / max Values				MM, MQ, MT, MV, XM, XQ, XT, XV									
Stamps set by the Stations	BL, CT, OR, RA, RB, SB, TM												
Post-processing	PO, EH												
Event Handling	AA, AE, AF, AH, AN, EE			TH									
Printout Handling	LD, PA, PF, PH, PU												
History Buffering	HE, HA, HF, HH, HL												
Historian logging	GN, GP	GN, GP	GN, GP	GN, GP	GN, GP	GN, GP	GN, GP	GN, GP	GN, GP	GN, GP			
Blocking Attributes	AB, HB, PB, UB, XB												
Operation Counting		CE, CL, CV, CO	CE, CL, CV, CO					CE, CL, CV, CO				CE, CL, CV, CO	
S.P.I.D.E.R. RTU Attributes			SE, SP						OF, EP				

Table continues on next page

Object Types		All Types	Predefined types										
Attributes		BI	BO	AI	AO	DI	DO	DB	PC	BS	FT	OE	NT
IEC Specific Attributes	OG, QL		TY		TY		TY						
OPC Event Attributes												AK, CK, CM, CN, CQ, ID, NS, OQ, QU, SE, SN, VA, VC, VM, VQ, VT	
File Transfer Attributes											DC, FF, FN, FP, ID, ST		
Topology Attributes	NO											CN, LE, LP, LV, LX, ND, NF, NM, NS, RO	
Miscellaneous Attributes	DX, FI, FX, GI, RI, RX												
Event History Attributes	CA, ED, EM, EQ, ET, EX, HD, HM, HQ, HT, MX, US												

5.1.6 Process object groups

Process objects of predefined types are regarded as components of process object groups, where all objects have the same name, and individual objects are identified by means of indices. Up to 65 535 related process objects of the predefined types can be given the same name. Different I/O signals of a motor control, for example, can be given the same object name. Process objects of several different predefined types can be included in the same group. Likewise, both real and process objects without process connection (see below) can be contained in the same group. Objects of user defined types cannot be included in a group.

Every object in a group is defined separately and independently of the other objects in the group. Individual attribute values are elements in a vector, formed by the corresponding attribute values of all objects in the group. Each attribute value is identified by the index of the object.

5.1.7 Storage

Process objects are stored in the process database file APL_PROCES.PRD. When the application is WARM or HOT, the process objects are stored in the global memory pool of SYS600 as well (see the System Objects manual).

All the attributes are stored in RAM. Depending on the switch state (see the attribute SS), the process object values (OV) are updated either both on disk and in RAM or only in RAM.

A process database may contain up to 2 000 000 process objects. Note, however, that a full size database occupies more than 1 GB of the primary memory. The memory pool should hence be configured accordingly.

At application start up, the process database is copied from disk to RAM. As the values stored on disk are probably outdated, the process database should be updated from the stations. This is managed differently for different types of stations. Example:

- S.P.I.D.E.R. RTUs send all object values (input data) automatically to the process database when the NET unit has been started (the NET unit has sent an SCI (Status Check Instruction)). When the NET unit is running, process database can be updated from the RTUs by setting the STAn:SSC attribute for all RTUs. For more information on this attribute see the System Objects manual. For example, the setting can be done in the initialization programs, which are application dependent command procedures started by the event channels APL_INIT_1 and APL_INIT_2. For more information on these event channels, see [Section 11](#).
- The process data of stations on ANSI lines (Allen-Bradley, SRIO, etc.) is read and updated in the process database by means of the #GET command ([Section 4.3](#)), for example, in the initialization programs.
- Process data of SPACOM units are updated in the process database when the STAn:SUP attribute of the stations (see the System Objects manual) is set. For example, the statement #SET STA2:SUP = 1 means that all process data of the SPACOM unit defined as STA2 is updated in the process database.

5.1.8 Process object notation

Process object attributes are accessed in SCIL with the following notation (see also [Section 4](#)):

name:[application]P[attribute]{(index)}

or

name:[application]P[attribute]{index}

where

'name'	is the name of the process Group (predefined types) or process object (user-defined types)
'application'	is the logical application number
'attribute'	is the attribute name
'index'	is an index or an index range

For predefined object types, the indices refer to individual objects in a group. An object notation without an index refers either to a process group attribute or to the process object with the lowest index, depending on the attribute. For user-defined object types, the indices refer to elements in user-defined attributes of vector type. In the following attribute descriptions ([Section 5.2](#) and [Section 5.3](#)), indexing is only explained in special cases.

Depending on the context, a process object notation without attribute name refers either to the default attribute OV (Object Value) or to the entire object. The object value attribute OV is an alias name for the main attribute of the object. Depending on the process object type, it refers to attribute BI, BO, AI, AO, DI, DO, DB, PC, BS, FT, OE or the main attribute of the user-defined object type. The entire object is referred by object handling commands, such as #CREATE, #DELETE, #MODIFY and #LIST.

The process data stored in stations using the ANSI X3.28 protocol can be directly accessed with the system object attribute STAn:SME (see the System Objects manual).

5.2 Configurable process object attributes

This section describes the process object attributes that define the objects and their functions. The attributes are grouped into the following sub-sections:

Section 5.2.1	Basic definition attributes:	IX, LN, PT, ZT
Section 5.2.2	Identification attributes:	CX, ES, IE, IL, OI, ON, OX, TX
Section 5.2.3	Addresses:	IG, IN, OA, OB, OT, TI, UN
Section 5.2.4	Operational state:	IU, SS, SU
Section 5.2.5	Scaling:	BC, DP, IR, SC, SN, ST
Section 5.2.6	Alarm handling:	AG, LA, NV, AC, AD, PD, PI, RC, WC, WR
Section 5.2.7	Limit value supervision:	HI, HO, HW, LI, LO, LW, SZ, ZD, ZE
Section 5.2.8	Post-processing:	EH, PO
Section 5.2.9	Event handling:	AA, AE, AF, AH, AN, EE, TH
Section 5.2.10	Logging of event history:	HA, HE, HF, HH, HL
Section 5.2.11	Historian logging:	GP, GN
Section 5.2.12	OPC event generation	
Section 5.2.13	Printout handling:	LD, PA, PF, PH, PU
Section 5.2.14	Network topology:	CN, NM, NO, NS, RO
Section 5.2.15	Miscellaneous attributes:	CE, CL, DX, FI, FX, GI, RI, RX, RZ

5.2.1 Basic definition attributes

5.2.1.1 IX Index

The index of a process object of predefined type. The individual objects in a group (up to 65 535 objects with the same name) are identified by indices.

In principle, the index for a new process object can be freely chosen. However, there are station type specific conventions that are widely used. As an example, the index of an event recording object in S.P.I.D.E.R. RTUs should be assigned the index of the supervised object plus 100.

Data type:	Integer
Value:	1 ... 65 535
Access:	Read-only, configurable

5.2.1.2 LN Logical Name

The logical name of the process group that the object belongs to. The individual process objects in the group are identified by indices (the IX attribute).

Data type:	Text
Value:	Object name
Access:	Read-only, configurable

Example:

Modifying the LN attribute, that is, moving the object from a group to another:

```
#MODIFY ABC:P1 = LIST(LN = "DEF")
```

5.2.1.3 PT Process Object Type

The type of the object. The process objects are grouped into twelve predefined process object types depending on the role of the object (input/output) and the type of the object value. In addition, up to 156 user defined types may be specified.

Data type:	Integer
Value:	1 ... 255
	Type numbers 1 ... 99 are reserved for predefined types.
	Type numbers 100 ... 255 can be used for user defined types.
	The predefined types are encoded as follows:
3	Binary Input (BI)
5	Binary Output (BO)
6	Digital Input (DI)
7	Digital Output (DO)
9	Analog Input (AI)
11	Analog Output (AO)
12	Double Binary Indication (DB)
13	Pulse Counter (PC)
14	Bit Stream (BS)
15	File Transfer (FT)
16	OPC Event (OE)
17	Network Topology
Access:	Read-only, configurable with #CREATE but not with #MODIFY

5.2.1.4 ZT Modification Time

The time when the object was created or modified. This attribute is automatically set when the object is created and each time it is updated by the #MODIFY command (for example, by the process object definition tool).

Data type:	Time
Access:	Read-only
Object types:	All

5.2.2 Identification attributes

5.2.2.1 CX Comment Text

A freely chosen text.

Data type:	Text
Value:	Any Unicode text, up to 255 characters
Default value:	""
Access:	No restrictions
Object types:	All

5.2.2.2 ES Event Source

The alias name of the object to be used as the event source name for the OPC Alarms & Events Server.

Data type:	Text
Value:	Up to 255 Unicode characters
Default value:	""
Access:	No restrictions
Object types:	All

The value of the attribute is a hierarchical name that consists of any number of fields separated by delimiters:

"area1.area2.arean.source"

Each field may contain any visible characters. The name is case sensitive. Single embedded spaces are allowed in a field, whereas leading and trailing spaces are not accepted.

By default, the characters recognized as delimiters are the colon (:), the slash (/), the backslash (\) and the dot (.). An application may define its own delimiter character set in the application attribute APL:BOP (OPC A&E Configuration), see the System Objects manual.

Examples of valid ES attribute values:

- "South Tipperary:Kilkenny.Relay1.Breaker.Position"
- "Äänekosken ala-asema:Laukaan syöttö/Katkaisija.Tila"

The event source name (ES) and the event handling object (EH) together identify the OPC event. Consequently, two or more process objects may have the same ES attribute value, provided that they are connected to different event handling objects.

The hierarchy formed by the ES attribute values is shown by the area browser of the OPC Alarms and Events Server.

5.2.2.3 IE Identifier Elements

This attribute shows the object's Object Identifier (attribute OI) divided into hierarchy levels.

Data type:	Vector, the maximum length used by the application is defined by APL:BOI
Element type:	Text
Element value:	The corresponding hierarchy level in the Object Identifier
Default value:	Empty vector
Access:	Read-only
Object types:	All

See also attributes IL and OI below.

Example:

```
If OI is
"STA1"           BAY2           DEV3"
then IE is
VECTOR("STA1", "BAY2", "DEV3")
```

when the default OI structure is used.

5.2.2.4 IL Identifier List

This attribute shows the object's Object Identifier (attribute OI) divided into named hierarchy levels.

Data type:	List, the attribute names correspond to the hierarchy level names defined by APL:BOI
Element type:	Text
Element value:	The corresponding hierarchy level in the Object Identifier
Default value:	Empty list
Access:	Read-only
Object types:	All

See also attributes IE above and OI below.

Example:

```
If OI is
"STA1"           BAY2           DEV3"
then IL is
LIST(STA = "STA1", BAY = "BAY2", DEV = "DEV3")
```

when the default OI structure is used.

5.2.2.5 OI Object Identifier

A freely chosen text that is used as a hierarchical location identifier for the object. The attribute is divided into sub-fields (hierarchy levels) according to the application specific conventions defined in the application attribute APL:BOI, see the System Objects manual.

Data type:	Text
Value:	Up to 63 Unicode characters
Default value:	""
Access:	No restrictions
Object types:	All

See also attributes IE and IL above.

5.2.2.6 ON OPC Item Name

The alias name of the object to be used as an OPC item id.

Data type:	Text
Value:	Up to 255 Unicode characters. For the syntax, see below.
Default value:	""
Access:	No restrictions
Object types:	All

The value of the attribute is a hierarchical name that consists of any number of fields separated by dots:

"field1.field2.fieldn"

Each field may contain any visible characters, except for colons and dots. The name is case-sensitive. Single embedded spaces are allowed in a field, whereas leading and trailing spaces are not accepted.

Examples of valid ON attribute values:

- "South Tipperary.Kilkenny.Relay1.Breaker.Position"
- "Äänekosken ala-asema.Laukaan syöttö.Katkaisija.Tila"

The names are unique within an application. Consequently, the user cannot, for example, give the same ON attribute value to a process object and a data object.

The hierarchy formed by the ON attribute values is shown by the name space browser of the OPC Server.

5.2.2.7 OX Object Text

A freely chosen text.

Data type:	Text
Value:	Up to 63 Unicode characters
Default value:	""
Access:	No restrictions
Object types:	All

5.2.2.8 TX Translated Object Text

The value of the OX attribute translated into the current language.

Data type:	Text
Value:	Any text
Access:	Read only
Object types:	All

When the attribute is read, the value of the OX attribute is taken as a text identifier and its translation into the current language is given. If the translation is not found, the translation into English is given. If no translation is found, the value of the OX attribute is returned as is.

5.2.3 Addresses

The attributes in this section specify the relationship between the process objects and the corresponding input and output signals and data in the process units.

For any process object, the UN attribute (Unit Number) specifies the unit where the corresponding process signal is located. Attributes OA (Object Address) and OB (Object Bit Address) specify the address of the signal within the unit.

Attribute IN (Item Name) specifies the name of the signal within the unit that uses names for signals instead of numeric addresses. Attribute IG (Item Group) is used to group items for communication with the data source (OPC Data Access Server).

5.2.3.1 IG Item Group

The name of the item group the item belongs to.

Data type:	Text
Value:	Any Unicode text up to 255 characters, case-sensitive
Default value:	""
Access:	No restrictions
Object types:	All types

An item group is a collection of items that share some communication properties defined by the OPC Data Access standard. This attribute has a meaning only if the data source of the object is an OPC Data Access Server.

If IG = "" (undefined), the default item group "DefaultGroupIn" or "DefaultGroupOut" is used, depending on the object type.

For output objects (object types BO, DO and AO), non-empty group names are not allowed. The default group "DefaultGroupOut" is always used.

The properties of item groups are specified in node (NODn:B) objects, see the System Objects manual.

The change of the item group name has no immediate effect on an active OPC connection. The groups are set up when the connection to the OPC server is established.

5.2.3.2 IN Item Name

The name of the signal within the process unit or communication engine.

Data type:	Text
Value:	Any Unicode text up to 255 characters, case-sensitive
Default value:	""
Access:	No restrictions
Object types:	All

An input object and an output object of the same data type (BI and BO, DI and DO, AI and AO) may share an IN attribute value. Otherwise, the item names are unique within a unit.

Named signals are typically used by process units that expose their data via an OPC Data Access Server or with communication engines which support string based addressing. With communication engine such as ICCP, the given IN acts as string based address instead of numerical OA Object Address. A matching address with the protocol configuration updates the process object if it is in automatic state, see attribute SS.

With OPC, the item names are often hierarchical names, where the dot character is used as a delimiter. The validity of the IN attribute is checked at run-time when the item is subscribed to and when data is received from the OPC Server. Any errors are reported via the event channel INVALID_OPC_ITEM, see [Section 11](#).

For OE (OPC Event) objects, this attribute defines the source of the event as specified by the OPC Alarms and Events specification. Several process objects may share an IN attribute value, because one source may generate different types of events. For details, see [Section 5.3.10](#).

5.2.3.3 OA Object Address

The address of the signal within the process unit. All real objects (the objects that are connected to a process signal) require an object address, unless they are addressed by the IN attribute. Possible bit address is given separately by the attribute OB.

For process objects belonging to stations using the ANSI X3.28 protocol, the object address is the same as the word address defined in the station.

For process objects belonging to S.P.I.D.E.R. RTUs and SPA units, the object address is a number coded according to the formula:

$$4096 * \text{object type number} + \text{logical address}$$

where

'object type number' is 0 ... 11 according to the following:

0 = No object type (simple input, counter input, measurand input in P214)

1 = Object command (S.P.I.D.E.R. RTUs), binary output (SPA)

2 = Regulation command (S.P.I.D.E.R. RTUs), command output (P214)

3 = Digital setpoint

4 = Analog setpoint

5 = General persistent output (S.P.I.D.E.R. RTU), setpoint output (P214)

6 = Analog value

7 = Indication (single or double)

8 = Pulse counter

9 = Digital value

10 = Indication event recording

11 = Analog event recording

'logical address' is the type specific logical address:
For S.P.I.D.E.R. RTUs the block address.
For SPA units: the SPA point address defined in NET.

The object address of process objects belonging to REX stations (REF, RED, REC, REL, etc., relays on a LON) is the address given in the process units.

The object address of process objects belonging to LMK stations (LSG device, Weidmuller, etc., on LON) is the address defined in NET by the corresponding LONWORKS^[1]

In the Process Object Definition Tool both the OA attribute and the logical address can be used.

Data type:	Integer
Value:	0 ... 2 147 483 647 0 = No object address.
Default value:	0
Access:	Read-only, configurable
Object types:	All

5.2.3.4 OB Object Bit Address

The bit address of the object in the station. The use of bit addresses is station type specific. Binary input, binary output and double binary process objects may have a bit address (see System Objects manual).

Double binary objects occupy two subsequent bit addresses: the bit specified by OB and the next bit.

Giving a bit address requires that the word address (the attribute OA) has been specified.

Because 0 is a valid value for OB, value 16 is used to indicate that bit addressing is not used.

Data type:	Integer
Value:	0 ... 16
Default value:	16 = No bit address
Access:	Read-only, configurable
Object types:	BI, BO and DB (value 16 for all other types)

5.2.3.5 OT Output Type

The representation - decimal, octal or hexadecimal - to be used when displaying the address (OA and OB) of the object.

[1] LONWORKS is a registered trademark of Echelon Corporation.

Data type:	Integer
Value:	0 Decimal representation
	1 Octal representation
	2 Hexadecimal representation
Default value:	0
Access:	No restrictions
Object types:	All

5.2.3.6 TI Table Index

Table index attribute supports configuring COM500i object addressing. Reserved for use by LIB 500.

Data type:	Integer
Access:	No restrictions
Object types:	All

5.2.3.7 UN Unit Number

The logical number of the station where the object is found (see the System Objects manual). Objects with a UN value of 1 ... 50 000 may be connected to the process through the communication system.

Data type:	Integer
Value:	0 ... 65 535
Default:	0
Access:	Read-only, configurable
Object types:	All

5.2.4 Operational state

5.2.4.1 IU In Use

State of use. This attribute determines whether the object can be operated or not. Taking an object out of use (IU = 0) means that all functions of the object, such as updating and alarm and event handling, are switched off. If the switch state is AUTO (SS = 2) or FICTITIOUS (SS = 3) when the object is taken into use (IU set to 1), the OS attribute (Object Status) gets the value 10 (NOT_SAMPLED_STATUS, see the OS attribute).

Data type:	Integer	
Value:	0	Out of use
	1	In use
Default value:	0	
Access:	No restrictions	
Object types:	All	

5.2.4.2 SS Switch State

This attribute describes how the object value attribute OV is updated: manually, automatically or not at all.

Data type:	Integer
Value:	0 ... 3
	0 OFF, no updating, the object is not used.
	1 MAN, manual updating. The object is updated by SCIL (#SET). The object has no connection to the process. The object value is stored on disk.
	2 AUTO, the object is connected to the process. The AI, BI, DI, PC and DB attributes can be updated from SCIL only if the object's UN = 0 and SS = 2. The object value is not stored on disk.
	3 FICTITIOUS. The object is fictitious. It is updated by SCIL (#SET). The object has no connection to the process. The object value is not stored on disk.
Default value:	0
Access:	No restrictions
Object types:	All

Table 3: The table shows how different switch states affect the reading and writing of the OV attribute. When the value is changed with a definition tool, it is changed according to the principles in the modify column.

	#SET obj:pov[ix] = value	#MODIFY obj:p[ix] = LIST.ov = value	OV (RAM)	OV (DISK)
SS=0	not possible, error 2013 (prof object switched off) is produced	changes DISK value	status 2013	defined
SS=1	changes RAM and DISK values	changes RAM and DISK values	defined	defined
SS=2	UN = 0: changes RAM value UN > 0: input objects >> not possible, error 2018 (prof update capability error) is produced output objects >> changes RAM value	UN = 0: changes RAM and DISK values UN > 0: input objects >> changes DISK value output objects >> changes DISK value	UN = 0: defined UN > 0: input objects >> status 10 output objects >> defined	defined
SS=3	changes RAM value	changes RAM and DISK values	defined	defined

5.2.4.3 SU Substitution State

The substitution state tells whether the value of the object (OV attribute) originates from the process, or is given by hand by the operator. Substitution mechanism is used to patch the process database by hand in case of a communication break or station hardware malfunction. If the operator knows the correct value of the object, he or she may write the value into the process database (along with the SU attribute value 1) to allow the applications that rely on the value work.

The value of SU is stored on disk. When SU is set to 1, the value of the OV attribute is also stored on disk, even if the switch state of the object is AUTO.

The attribute may be set by a simple #SET command or a list type #SET command may be used to set the SU attribute and the corresponding OV attribute at the same time, for example:

```
#SET X:P1 = LIST(SU = 1, BI = 1)
```

The attribute is automatically reset from 1 to 0, when the value of the OV attribute is updated by the process (SS = AUTO) or by SCIL (other switch states).

Setting the SU attribute generates an event and activates an event channel, printout and history logging, if enabled.

Data type:	Integer	
Value:	0	Not substituted
	1	Substituted
Default value:	0	
Access:	No restrictions	
Object types:	All	

5.2.5 Scaling

5.2.5.1 BC Bit Count

The number of bits in the maximum value of the pulse counter. This is an informative attribute that entails no automatic functions in the process database, but can be used in SCIL programs.

Data type:	Integer
Value:	1 ... 32
Default value:	16
Access:	No restrictions
Object types:	PC

5.2.5.2 DP Decimal Places

Number of decimal places used to display the value of the object.

Data type:	Integer
Value:	-1 ... 10
Default value:	-1 (= unspecified)
Access:	No restrictions
Object types:	AI and AO

5.2.5.3 IR Integer Representation

The representation of analog values.

When IR = 1, the following attributes are represented as integer values:

AI objects: AI, MV, XV, LI, HI, LW, HW, ZD

AO objects: AO, LO, HO

When IR = 0, the attributes are represented as real (floating point) values.

Furthermore, when IR = 1, the value is not scaled between the database and the station. The scale name attribute SN must be empty, if given at all, when the object is created.

Data type:	Integer	
Value:	0	Floating point representation. Data type REAL in SCIL
	1	32-bit signed integer representation. Data type INTEGER in SCIL
Default value:	0	
Access:		Read-only, configurable. It can be set with the #CREATE command, but it cannot be modified with the #MODIFY command.
Object types:	AI and AO	

5.2.5.4 SC Scaling

The scaling of a pulse counter. It indicates the number of units that one pulse corresponds to. The unit is determined by the ST (Engineering Unit) attribute.

The attribute has only an informative function that may be used by SCIL.

Data type:	Real
Value:	Size of one pulse
Access:	No restrictions
Object types:	PC

Example:

```
ENERGY = COUNTER:PPC3 * COUNTER:PSC3
```

5.2.5.5 SN Scale Name

The name of the scale used for the scaling of the object. A scale is an algorithm for converting between the representation of analog data in the process unit and in the database, in accordance with the engineering unit of the object (the ST attribute). Every real value (see attribute IR) analog process object must have a scale (see [Section 7](#)). The scale must exist in the database before the process object can be created.

Scaling is done in the process database before the object value is registered (AI) or sent to the process (AO).

Data type:	Text
Value:	Object name
Access:	Read-only, configurable
Object types:	AI and AO

5.2.5.6 ST Engineering Unit

The engineering unit of the object value.

Data type:	Text
Value:	Up to 10 characters
Access:	No restrictions
Object types:	AI, AO and PC

5.2.6 Alarm handling

5.2.6.1 General

Process objects of any type can be equipped with alarm handling. The following object types have alarm generation that is based on the object value (OV):

- Analog input objects (AI).
- Binary input objects (BI).
- Double binary objects (DB).
- OPC event objects (OE).
- Objects of user-defined types.

Control supervision alarms are alarms that are generated when control of a device fails, that is, indication of the expected new state is not received within a specified period of time. Control supervision alarms are independent of value based alarms. Control supervision is described in more detail in section Process object group attributes.

In addition, objects of any type may generate alarms that are based on the status of the object, see [Section 5.2.6.2](#).

Alarms are categorised into 7 **alarm classes** (attribute AC, see below). The base system software does not make any distinction between the classes, but applications may use alarm classes to categorise alarms, for example according to their severity. AC attribute value 0 disables all alarm handling of the object.

High Warning (HW) and Low Warning (LW) limits can be configured to generate alarms for analog input objects. Alarms generated by low warning or high warning have their own alarm class (attribute WC, see below).

The **alarm list** contains all the alarming objects of the application. The list is maintained in time order (the AT and AM attributes, see [Section 5.3.3](#)). An object may appear in the list only once. Obligatory acknowledgement of alarms may be specified for a process object (attribute RC, see below). In this case, the alarming object stays on the list until the alarm is acknowledged even if it is updated to a non-alarming value.

An **alarm picture** to be shown on one or more **alarm monitors** may be associated to the alarm (see attributes PD and PI below). When an alarm is generated, the name of the alarm picture is appended to the monitor specific alarm picture queue. At the same time a monitor alarm signal (a red flashing square in the upper right corner) is displayed. The command !INT_PIC shows the alarm picture that is first in the queue. When the picture is shown, it is removed from the queue. The alarm picture stays in the queue until it is shown and clearing the alarm does not remove it from the queue. When a semi-graphic monitor is closed, the remaining alarm pictures in the alarm picture queue are shown one by one. If an alarm occurs while the semi-graphic monitor is closed, the alarm picture is automatically and immediately displayed on the screen.

By default, changes of alarm state (on/off) are considered as events by SYS600. Consequently, post-processing of events (printouts, event channel activation and history logging, see [Section 5.2.8](#)) applies to alarms as well.

Alarm handling may be temporarily disabled by setting the **alarm blocking** attribute AB to 1 (see [Section 5.3.4](#)).

If an object has an **alarm delay** (non-zero AD attribute value, see below), a short alarm state can pass without alarm activation.

Every time an alarm is generated, event channel named APL_ALARM is activated (see [Section 11](#)) and application attribute AT (Alarm Tag, APL:BAT) is incremented to notify the application

about the alarm. Active alarms by alarm classes are counted by APL:BAC and unacknowledged alarms by APL:BUC.

The attribute GA (Group Alarm) of the process object group indicates, whether any of the objects of the group are in alarming state, see [Section 5.5](#).

If an audio alarm unit is connected to the system, an audio alarm for the alarm class of the process object is set whenever an alarm is generated.



In MicroSCADA revisions prior to 8.4.5 SP2 and 9.1, an audio alarm was generated only when an alarm row was printed. Use the application revision compatibility switch "COUPLE_AUDIO_ALARMS_AND_PRINTOUTS" (see the System Objects manual) to keep this obsolete feature while upgrading to a newer revision.

Alarm handling is roughly illustrated in [Figure 5](#).

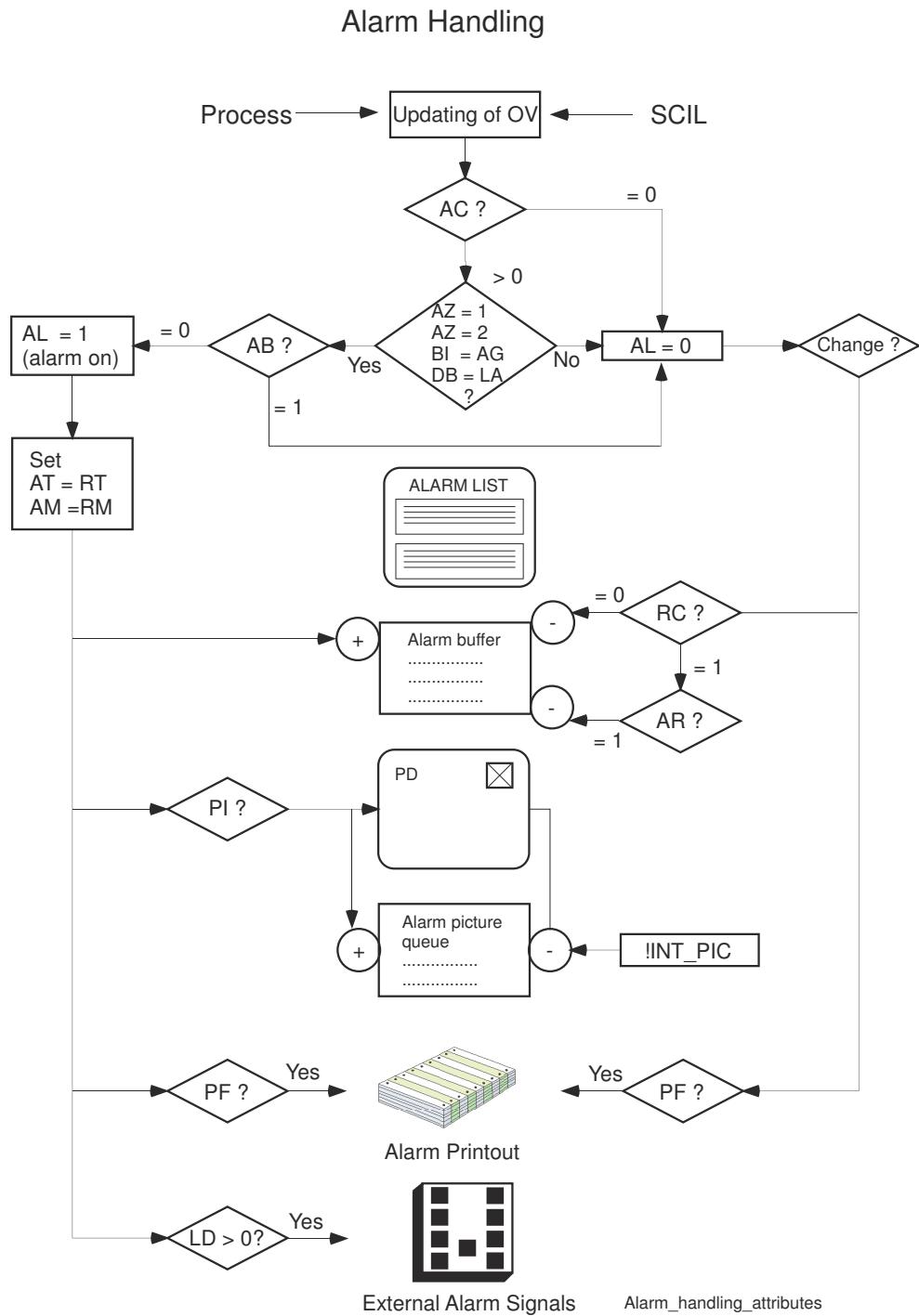


Figure 5: A rough outline of the alarm handling attributes for analog input and binary input objects. OS is supposed to be 0. If the old OS = 10 the monitor alarm, alarm printout and external alarm signals depend on the PU attribute (Picture at First Update, see [Section 5.2.13](#)).

5.2.6.2 Alarm generation

Alarm generation is based on the value and the status of an object.

Provided that the object has an alarm class (AC > 0) and the alarms are not blocked (AB = 0), an alarm is generated in the following cases:

- The value of an analog input object (or an integer or real valued object of a user-defined type) enters the low alarm or high alarm zone (AZ attribute, see [Section 5.2.7](#)). The value of AZ is either received from the station (SZ = 0) or it is calculated by SYS600 (SZ = 1). In the latter case, the alarm zones are specified by the alarm limit attributes HI and LI.
- A binary input object (or a Boolean object of a user-defined type) gets the alarm generating value (the AG attribute). At the first update, the alarm can be prevented by the NV attribute (see below).
- A binary double indication gets any of the alarming values (the LA attribute). At the first update, the alarm can be prevented by the NV attribute. See below.
- An OPC event object that represents an instance of an OPC condition enters the Activated state, that is, the OE attribute changes from 0 to a non-zero value.
- The OS attribute (Object Status, see [Section 5.3.2](#)) gets the value 1.
- The OR (Out of Range, see [Section 5.3.7](#)) or OF (Overflow, see [Section 5.3.8](#)) attribute gets value 1.
- Analog input object is configured to generate an alarm (WC > 0) when low warning or high warning condition is entered.

Another type of alarms, control supervision alarms, are generated when a control fails to change the state of the controlled device to the expected value within a given period of time. Control supervision alarms are defined by the process object group attribute CC (Control Supervision Configuration).

For binary and double binary objects, alarm generation is specified by the following attributes:

AG Alarm Generation

Alarm generation of binary input objects (BI). This attribute specifies which bit value, 0, 1 or both, will generate an alarm. When the BI attribute gets this value, an alarm is generated. If both bit values generate an alarm, the alarm generation at the first update can be prevented by means of the Normal Value (the NV attribute, see below).

The attribute is also valid for user defined object types of data type Boolean.

Setting AG to 0 or 1 may generate or clear an alarm according to the current value of the BI attribute, unless prevented by the revision compatibility (APL:BRC) setting "SETTING_LA_AND_AG_DOES_NOT_ALARM".

Data type:	Integer	
Value:	0	Bit value 0 generates an alarm
	1	Bit value 1 generates an alarm
	2	Both bit value 0 and bit value 1 generate an alarm
	3	Both bit values 0 and 1 generate an alarm on each update (even if the value hasn't changed)
	4	Bit value 0 generates an alarm on each update
	5	Bit value 1 generates an alarm on each update
Default value:	1	
Access:	No restrictions	
Object types:	BI and user-defined types	



If AG equals 2, 3, 4 or 5, the alarm is removed when it is acknowledged.

LA Alarm Activation

Alarm generation of double binary indications (DB), given as a bit mask. Several states, even all four states can be alarm generating.

If one, two or three states are defined to be alarm generating, an alarm is generated when the object value (the OV attribute) receives any of these states. The alarm disappears when a non-alarming state is received. If all four states are defined to generate an alarm (LA = 15), each new update causes a new alarm. In this case, the alarm disappears when it is acknowledged.

When an object with LA = 15 is updated for the first time (OS goes from 10 to 0) and the object value is the normal value (NV), or the normal value is 4, no alarm is generated (see the NV attribute).

Setting LA (to a value other than 15) may generate or clear an alarm according to the current value of the DB attribute, unless prevented by the revision compatibility (APL:BCR) setting "SETTING_LA_AND_AG_DOES_NOT_ALARM".

Data type:	Integer
Value:	0 ... 15. The alarm generating states given as a bit mask 0 = No alarm generation
Default value:	0
Access:	No restrictions
Object types:	DB

Example:

The DB values 0 and 2 will be generate an alarm (the value of DB:PLA1 value will be 5):

```
#SET DB:PLA1 = BIT_MASK(0,2)
```

LA Alarm Activation

Alarm generation of network topology objects (NT), given as a bit mask. Several states, even all states can generate an alarm.

An alarm is generated when the object value (NT attribute) receives any of the states defined in the LA attribute. The alarm disappears when a non-alarming state is received. For more information about network topology objects, see [Section 5.2.14](#).

Setting LA may generate or clear an alarm according to the current value of the NT attribute, unless prevented by the revision compatibility (APL:BCR) setting "SETTING_LA_AND_AG_DOES_NOT_ALARM".

Data type:	Integer
Value:	0 ... 63. The alarm generating states given as a bit mask. In the POWER schema, the NT attribute may have values 0 ... 5. 0 = No alarm generation
Default value:	0
Access:	No restrictions
Object types:	NT

Example:

The NT values 1 (Unpowered) and 2 (Uncertain) will be generate an alarm (the value of NT:PLA1 value will be 6):

```
#SET NT:PLA1 = BIT_MASK(1,2)
```

NV Normal Value

The normal value of a binary input or double binary object.

The NV attribute is applied only at the first update after the application start-up and when all states are alarm generating. All states are alarm generating when AG = 2 for binary input objects or LA = 15 for DB objects. In these cases, the object does not cause any alarm if its value is the same as the NV attribute or if NV = 2 (BI objects) or NV = 4 (DB objects).

Data type:	Integer
Value:	0, 1 or 2 for binary input (BI) objects 0 ... 4 for double binary indications (DB)
Default value:	2 for binary objects and 4 for double indications. These values mean that no alarm is generated at start-up for objects that are defined to be alarm generating in all states.
Access:	No restrictions
Object types:	BI and DB

Example:

If a binary input object defined to generate alarm in all states (AG = 2) has NV = 1, no alarm is generated if BI = 1 at the first update. However, if BI = 0 at first update, an alarm is generated. If the NV attribute of the object = 2, no alarm is generated at first update regardless of the value of BI.

5.2.6.3 Alarm handling attributes

The following attributes specify the alarm functions of the object:

AC Alarm Class

There are seven equally significant alarm classes for grouping alarms. The application engineer chooses how to group the objects in alarm classes. The objects can, for example, be grouped based on the location of the process objects or alarm severity. An object with alarm class 0 has no alarm function.

Changing AC from non-zero value to 0 clears the alarm indication (AL attribute) of the object. Changing AC from 0 to a non-zero value does not affect the alarm indication, AL remains as 0.

When AC is changed, an event is generated (if EE = 1) and the change is registered in the history database (if HE = 1).

The alarm class is of significance when connecting the alarms to audio or audiovisual alarm signals through additional circuit boards. All objects belonging to the same class have the same type of audio alarms.

Data type:	Integer
Value:	0 ... 7 0 = No alarm function
Default value:	0
Access:	No restrictions
Object types:	All

AD Alarm Delay

Alarm delay specifies the delay between the registration of an alarming value in the process database and responding to the alarm. If an alarm delay is specified, the alarming value is updated in the process database normally (attributes OV, OS, RT RM, RQ, AT, AM, AQ, YT, YM and YQ) are updated), but the post-processing of the alarm is postponed. In this case, the AL is not set, the object is not included in the alarm list and no other post-processing that depends on the alarm state is done. When the delay expires, the AL attribute (Alarm, see [Section 5.3.3](#)) is set to 1, attributes RT, RM and RQ are updated and the alarm post-processing

is done, provided that the alarming value still remains. If the object value is updated to a non-alarming value during the delay, the pending alarm is cancelled.

The alarm delay is applied only when an alarm is about to be raised because of an alarming OV value. Alarms caused by bad status of the object (such as OS value FAULTY) are raised immediately. Also, clearing of an alarm takes place immediately, alarm delay is not applied.

For analog input objects, the warnings are also delayed according to the AD attribute value. If the object value is updated to an alarming value during a warning delay, the delay is restarted. If the value returns to normal during the alarm delay, neither alarm nor warning is produced. However, if the value returns to a warning state, a warning is immediately generated.

Data type:	Integer
Value:	0 ... 65 535
Unit:	Seconds
Default value:	0
Access:	No restrictions
Object types:	All

PD Picture Devices

The logical monitor numbers of the alarm monitors, that is, the monitors where the picture alarm message and the alarm picture will be shown. An object can have up to 15 alarm monitors (the monitors with logical numbers 1 ... 15, see the System Objects manual).

Using the PU attribute (Printout at First Update, see [Section 5.2.13](#)), alarm messages, including the monitor alarm, can be inhibited when the object is updated for the first time.

Data type:	Integer
Value:	0 ... 65 534, even numbers. The logical monitor numbers of the alarm monitors given as a bit mask.
Access:	No restrictions
Object types:	All

Example:

Monitors by logical numbers 2 and 4 will receive monitor alarms:

```
#SET A:PPD = BIT_MASK(2, 4)
```

PI Picture

The name of the alarm picture of the object.

Data type:	Text
Value:	Picture name
Access:	No restrictions
Object types:	All

RC Receipt

Demand for acknowledgement. The attribute states whether acknowledgement of the alarm is obligatory or not. If acknowledgement is required, the alarm is not removed from the alarm list until it is acknowledged (by setting the AR or WR attribute).

Data type:	Integer	
Value:	0	No acknowledgement
	1	Demand for acknowledgement
Default value:	0	
Access:	No restrictions	
Object types:	All	

WC Warning Alarm Class

Warning alarm class follows the principles of the alarm class (AC) attribute. There are seven equally significant warning alarm classes for grouping. An object with alarm class 0 has no alarm function. This applies also to alarm generation for low warnings and high warning limits. When setting the WC value, the value of AC must be non-zero.

When WC is changed, an event is generated (if EE = 1) and the change is registered in the history database (if HE = 1).

Data type:	Integer		
Value:	0 ... 7	0 = No alarm function for low warning and high warning limits	
Default value:	0		
Access:	No restrictions		
Object types:	AI		

WR Warning Alarm Receipt

Warning alarm receipt follows the principles of the alarm receipt (RC) attribute. When setting the WR value, the value of RC must be 1.

The attribute states whether acknowledgement of the alarm is obligatory or not. If acknowledgement is required, the alarm is not removed from the alarm list until it is acknowledged (by setting the AR attribute).

Data type:	Integer		
Value:	0	No acknowledgement	
	1		
Default value:	0		
Access:	No restrictions		
Object types:	AI		

5.2.7 Limit value supervision

These attributes apply to analog input and output objects (AI and AO).

Attributes LI, LW, HW and HI (Low Input, Low Warning, High Warning and High Input) specify the alarm and warning limits of analog input objects. These attributes are also valid for user-defined objects of data types real and integer. The values of the attributes must obey the inequality LI <= LW <= HW <= HI. Setting a warning limit equal to the corresponding alarm limit effectively disables the warning. Setting all the limits to same value disables both alarms and warnings. If a warning and an alarm limit have been set equal (HI = HW or LI = LW) and the alarm limit is changed, the value of the warning limit will follow.

The dynamic attribute AZ (Alarm Zone) reflects the position of the value of AI attribute relative to the warning and alarm limit attributes. Whenever the alarm zone changes, the alarm state of

the object is re-calculated and post-processing is done. When any of the limit attributes is changed, the alarm zone is re-calculated.

Attributes LO and HO (Low Output and High Output) define the range of allowed AO attribute values for an analog output object. If HO = LO, no range checking is done.

Zero deadband supervision (attributes ZD and ZE) is used to filter out small disturbances and calibration errors of measurements.

5.2.7.1 HI High Input

The upper alarm limit of the analog input value. An alarm is raised (AL = 1), when the AI attribute exceeds this value (provided that SZ = 1).

Data type:	Real, if IR = 0
	Integer, if IR = 1
Default value:	0
Access:	No restrictions
Object types:	AI and user-defined types

5.2.7.2 HO High Output

The upper limit of the analog output value. Attribute AO cannot be set to a value larger than HO.

Data type:	Real, if IR = 0
	Integer, if IR = 1
Default value:	0
Access:	No restrictions
Object types:	AO

5.2.7.3 HW High Warning

The upper warning limit of the analog input value.

Data type:	Real, if IR = 0
	Integer, if IR = 1
Default value:	The value of HI attribute
Access:	No restrictions
Object types:	AI and user-defined types

5.2.7.4 LI Low Input

The lower alarm limit of the analog input value. An alarm is raised (AL = 1) when the AI attribute goes below this value (provided that SZ = 1).

Data type:	Real, if IR = 0 Integer, if IR = 1
Default value:	0
Access:	No restrictions
Object types:	AI and user-defined types

5.2.7.5 LO Low Output

The lower limit of the analog output value. Attribute AO cannot be set to a value smaller than LO.

Data type:	Real, if IR = 0 Integer, if IR = 1
Default value:	0
Access:	No restrictions
Object types:	AO

5.2.7.6 LW Low Warning

The lower warning limit of the analog input object value.

Data type:	Real, if IR = 0 Integer, if IR = 1
Default value:	The value of LI attribute
Access:	No restrictions
Object types:	AI and user-defined types

5.2.7.7 SZ SCADA Zone Supervision

This attribute determines whether the warning and alarm limit supervision is done by the station or by SYS600. When done by the station, the alarm and warning state of the object (attribute AZ, Alarm Zone) is sent along with the object value in the update message. In this case, attributes HI, LI, HW and LW are informative only. For availability to the SCIL application, they should be set according to the limits used by the station.

For S.P.I.D.E.R. RTU objects the SZ attribute should normally be 0 (supervision in the RTU).

Data type:	Integer (AI object type) or Boolean (user-defined object types)	
Value:	0 or FALSE	Supervision in the station
	1 or TRUE	Supervision in SYS600
Default value:	0 or FALSE	
Access:	No restrictions	
Object types:	AI and user-defined types	

5.2.7.8 ZE Zero Deadband Supervision Enabled

Analog input objects can be defined with zero deadband supervision. If supervision is enabled and the value of the object lies within the deadband, exact zero is stored as the object value (AI). As long as the value lies within the deadband zone, the measured entity (for example

current, voltage) is regarded as switched off and the variations are regarded as negligible disturbances.

When zero deadband is in use, the value 0 is not considered to be an alarming value even though a lower alarm limit $LI \geq 0$ is given. When the object gets the value 0, the alarm zone attribute AZ is set to 0. The Minimum Value attribute (MV) is not updated.

Data type:	Integer
Value:	0 No zero deadband supervision
	1 Zero deadband supervision is enabled
Default:	0
Access:	No restrictions
Object types:	AI

5.2.7.9 ZD Zero Deadband

The width of the zero deadband, see [Figure 6](#). The value of the object is regarded as zero when it lies within the shaded zone, that is, within the range $-ZD \dots +ZD$. See also the ZE attribute above.

Data type:	Real, if IR = 0
	Integer, if IR = 1
Value:	Width of the zero deadband
Default:	0
Access:	No restrictions
Object types:	AI

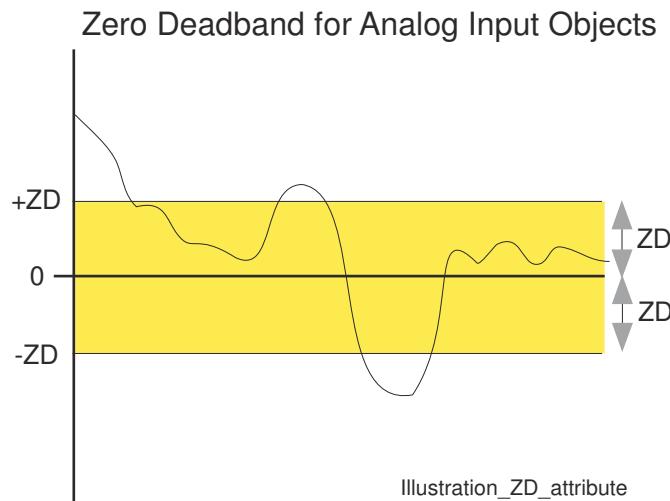


Figure 6: An illustration of the ZD attribute

5.2.8 Post-processing

5.2.8.1 General

When a process object is updated, several types of application specific actions may want to be taken. Alarm handling and limit value supervision were described in previous sections. The

others, collectively called event post-processing, or post-processing for short, are described in this and the three following sections.

The following types of post-processing are supported:

- Event handling: activation of event and event channel objects, see [Section 5.2.9](#)
- Logging of event history, see [Section 5.2.10](#)
- OPC event generation, see [Section 5.2.12](#)
- Printout handling, see [Section 5.2.13](#)

Mechanisms common to all or several of these types are described in this section.

The following events in the process object may initiate the post-processing:

- Attribute OV (Object Value) is changed (or only updated)
- Attribute OS (Object Status) is changed
- Attribute SE (Selection) or SP (Stop Execution) is set
- Attribute AZ (Alarm Zone) is changed
- Attribute SU (Substitution State) is changed
- A user defined attribute is changed (or only updated)
- Attribute AL (Alarm) is updated (see attribute PO below). However, if AL changes because of Alarm Blocking (AB) setting, post-processing is not done.
- Attribute AS (Alarm State) is changed. However, if AS changes because of Alarm Blocking (AB) setting, post-processing is not done.
- Attribute AR (Alarm Receipt) is set to 1, that is, an alarm is acknowledged
- Any of the protocol specific quality attributes BL (Blocked), SB (Substituted), TM (Test Mode), OR (Out of Range) or OF (Overflow) is changed

In addition, event object and history logging is activated also by a number of other attributes, see [Section 5.2.9](#) and [Section 5.2.10](#).

One process event may initiate multiple post-processing based on different changed attributes: The change of the OV attribute along with its immediate consequences (attributes OS, AZ, AL, AS) is handled in the first post-processing, followed by post-processing of each changed protocol specific quality attribute (BL, SB, TM, OR and OF). Event channel command procedures should check the snapshot variable CA (Changed Attribute) to find out the reason for the activation and act accordingly.

When any of the events listed above occurs, there are a few more conditions to be fulfilled before the actual post-processing takes place:

- Post-processing is enabled. This is specified by post-processing type specific attributes that are described in the following sections.
- The event is strong enough. This is defined by the activation criterion of the post-processing type, see [Section 5.2.8.4](#) below.
- Special treatment of the first event after the application start-up may filter out some events, see [Section 5.2.8.5](#) below.

Post-processing in case of alarm state update is defined by the PO attribute, see the description below.

Some additional definitions associated with the post-processing are defined by the following EH attribute.

5.2.8.2 EH Event Handling

The name of the event handling object associated with the process object.

Data type:	Text
Value:	The name of the associated event handling (EH) object
Default value:	According to the process object type
Access:	No restrictions
Object types:	All

This attribute holds a reference to the event handling object that stores some additional definitions for the post-processing. Currently, the values of the language-sensitive attributes SX (State Text) and MX (Message Text) are obtained via the referenced event handling object.

For OE (OPC Event) objects, the event handling object additionally defines the OPC events that are received by the object.



EH attribute is not implemented for user-defined process object types.

5.2.8.3 PO Post-processing Based on Object Value

In standard process event semantics, change of alarm state is considered as an event, even an especially “strong” one. Consequently, change of alarm state triggers event channel, printout and history database activation regardless of the corresponding activation criterion attribute (AA, HA and PA) value, unless the corresponding post-processing is totally disabled.

However, this is not the desired behavior in all applications. Especially, there are applications that do not want DOWN events to appear in the event list even if these events clear the alarm state.

PO attribute is used to select the semantics.

Data type:	Integer	
Value:	0	Post-processing is based on both OV and AL
	1	Post-processing is based on OV only
Default value:	0	
Access:	No restrictions	
Object types:	All	

5.2.8.4 Activation criterion

For each type of post-processing, there is an attribute (AA, PA and HA) that specifies the criterion for the activation, that is, which type of change of the value is required to trigger the activation.

The criterion may apply to:

- The OV attribute of the object (and the following OV related attributes: OS, SE and SP).
- The OV attribute and the acknowledgement of an alarm.
- The alarm state (AS attribute) of the object.

In the first case, the criterion normally applies to the OV attribute of the object. When the activation is caused by the change of some other attribute, the criterion is examined as if the OV value had been changed (NEW VALUE). However, if the alarm state is cleared by acknowledging the alarm (in case of a binary input or a double binary object, whose all states are alarming states), neither event channel nor printout is activated.

When the object status OS changes, the status value 2 (INVALID, or OBSOLETE) has a special treatment:

- When the object status is set to 2 as a consequence of the broken connection to the process station, no activation is done regardless of the activation criterion. The reason for this is that thousands or tens of thousands objects may be affected by the communication break.
- When the object status is explicitly set to 2 by SCIL or a process message, the activation is done if the criterion is NEW VALUE or UPDATE.
- When the object status returns from 2 to its normal value 0 (typically after the connection has been re-established) and the object value OV does not change, the activation is done only if the criterion is UPDATE.

The behavior in case of OS changes may be customised by the PP (Post-processing Policy for Object Status 2) attribute of the station and/or application object, see the System Objects manual.

Below, the criteria based on the object value are listed in the order of precedence. For example, if the alarm zone of an analog input object is changed from the normal to a warning zone or vice versa, the activation takes place if the criterion is UPDATE, NEW VALUE or WARNING. If the criterion is ALARM, activation is not done.

0	ALARM	Activation only when the AL attribute changes (an alarm comes or goes).
3	WARNING	Activation when the alarm zone (AZ) of an analog input object changes from the normal zone to a warning zone or vice versa, see Section 5.3.3 .
4	UP	Activation when a binary object (BI or BO) or a double binary object (DB) is changed 'up' or 'down'.
5	DOWN	For binary objects, 'up' means 0 -> 1 and 'down' 1 -> 0. For double binary objects, 'up' means 1 ->2 and 'down' 2 -> 1. The 'forbidden' transitions of DB objects (from and to states 0 and 3) cause both UP and DOWN activation.
1	NEW VALUE	Activation each time the value is changed.
2	UPDATE	Activation each time the value is updated (even if it is not changed). However, if the cause of transmission (CT) of the update is INTERROGATED and the first update attribute (AF, HF or PU) is 0, activation is not done. In this case, the update does not contain any new information.

The criteria based on the object value and the alarm acknowledgement are listed below according to precedence:

10	ALARM or ACK	Activation only when the AL attribute changes (an alarm comes or goes) or the alarm is acknowledged.
13	WARNING or ACK	Activation when the alarm zone (AZ) of an analog input object changes from the normal zone to a warning zone or vice versa, or the alarm is acknowledged, see Section 5.3.3 .
14	UP or ACK	Activation when a binary object (BI or BO) or a double binary object (DB) is changed 'up' or 'down' or the alarm is acknowledged.
15	DOWN or ACK	For binary objects, 'up' means 0 -> 1 and 'down' 1 -> 0. For double binary objects, 'up' means 1 ->2 and 'down' 2 -> 1. The 'forbidden' transitions of DB objects (from and to states 0 and 3) cause both UP and DOWN activation.
11	NEW VALUE or ACK	Activation each time the value is changed or the alarm is acknowledged.
12	UPDATE or ACK	Activation each time the value is updated (even if it is not changed) or the alarm is acknowledged. However, if the cause of transmission (CT) of the update is INTERROGATED and the first update attribute (AF, HF or PU) is 0, activation is not done. In this case, the update does not contain any new information.

The criterion based on the alarm state (AS attribute) of the object:

6	ALARM STATE	Activation only when the AS attribute changes.
---	-------------	--

5.2.8.5 First update

When an application starts up, the input type process objects that are connected to a station are marked with NOT_SAMPLED_STATUS, the OS (Object Status) attribute is set to 10. Normally, the stations are then asked to send the values of all their signals to SYS600, a General Interrogation is done. The interrogation results in a large number of process object updates in the process database. These updates are distinguished by the Cause of Transmission of the update. Its value is INTERROGATED instead of SPONTANEOUS which is used by consecutive updates sent spontaneously by the station.

Most of these start-up updates only initialize the process object to its current value and do not represent any event in the station. Therefore, no post-processing should be done.

However, there are applications that want to print or log the start-up value of some specific process objects or handle them by SCIL in a way or another. To enable this kind of application dependent tailoring, there is an At First Update attribute for each type of post-processing (attributes AF, PU and HF). Value 1 of one of these attributes specifies that the corresponding post-processing is done at the first update. The default value is 0, no post-processing at the first update.

When a SPONTANEOUS update is received from the station, the post-processing is done regardless of the At Fist Update attribute value even if the status of the process object is NOT_SAMPLED_STATUS before the update.

5.2.8.6 Snapshot variables

When an event channel (attribute AN) or a printout (format picture, attribute PF) is activated by the process database, the snapshot variables are passed as arguments to the activated object. SCIL code run by the activated object may access these variables as normal global variables.

Most of the variables are included in the set in order to describe the present state of the process object, that is, to take a snapshot of the dynamically changing object. These variables have the same name as the process object attribute they represent. The others describe the reason for the activation.

The following variables (by process object type) are included in the snapshot variable set. Each variable has the value of the attribute by the same name.

- All types: LN, IX, OV, OS, RT, RM, RQ, SX, AL, AS, BL, SB, TM, OR, OF, RA, RB, PB, XB, HB, UB, CT, OG, QL, TY and SU
- All alarming objects (AC > 0): AR, AT, AM, AQ, KT, KM, KQ, YT, YM and YQ
- Binary Input: BI
- Binary Output: BO, SE and SP
- Double Binary: DB
- Digital Input: DI
- Digital Output: DO
- Analog Input: AI, AZ, MV, MT, MM, MQ, WQ, XV, XT, XM and XQ
- Analog Output: AO
- Pulse Counter: PC and EP
- Bit Stream: BS
- File Transfer: FT, FF, FN, ID, DC and ST
- OPC Event: OE, AK, CK, CM, CN, CQ, ID, NS, OQ, QU, SE, SN, VC, VM, VQ, VT
- Network Topology: NT, LE, LP, LV, LX, ND, NF
- User defined types: Any user defined attribute specified to be included.

The following four variables tell the reason for the activation:

- CA (Changed Attribute) contains the name of the attribute whose change caused the activation. Most often CA equals to the name of the object value attribute ("BI", "AI" etc.), but also the following attributes may be the cause: OS, SB, TM, BL, OR, OF or SU.
- US (User Name) is the name of the MicroSCADA user who caused the event.
- MX (Message Text) is a textual description of the event in the language of the application.
- CHANGE tells the type of change that has happened:

"NONE"	The value has been updated but it has not changed
"VALUE"	The value has changed
"ZONE"	The alarm zone (of an analog input object) has changed
"ALARM"	The alarm state has changed

5.2.9 Event handling

This section describes the activation of event channel objects and event objects as a consequence of a process object update.

5.2.9.1 Event channel activation

Event channels are used to link a process object event to the report database objects, i. e. data and command procedure objects. In these objects, application specific post-processing of the event is done by SCIL programming. For more information on event channels, see [Section 10](#).

Process object updates that cause an activation of the event channel are described in [Section 5.2.8](#). The same section also describes the snapshot variables passed as arguments to the event channel, as well as some other functionality common to all types of post-processing.

To lower the load of event channel activations, so called threshold mechanism is implemented for analog input objects (see ["TH Threshold"](#)).

The activation of the event channel may be temporarily blocked by means of the Blocking Attributes (attribute XB, see [Section 5.3](#)).

5.2.9.2 Event activation

Event activation is used to implement event based updating of process pictures and other user interface objects.

When certain changes in the process object take place (see below), an event object by the name and index of the process object is activated. Event specific SCIL programs for this particular event in any of the open monitors are then executed. The event program is defined by #ON command, which can be used in any user interface object, or as an event method available in Visual SCIL objects. For more information on event objects, see [Section 12](#).

A change of any of the following process object attributes causes an event object activation:

AB, AC, AG, AR, AZ, BL, HB, HI, HO, HW, IU, LA, LI, LO, LW, OF, OR, OS, OV, PB, RC, SB, TM, SE, SP, SS, UB, XB, WC, WR and user defined attributes.

For attributes PC, RC, SE and SP, any update causes the activation of an event object, even if the value does not change.

5.2.9.3 Event handling attributes

AA Action Activation

This attribute specifies the value changes that activate the event channel. For details of the activation criteria, see [Section 5.2.8](#).

Data type:	Integer
Values:	0 ... 15, the activation criterion, see Section 5.2.8
Default value:	0
Access:	No restrictions
Object types:	All

AE Action Enabled

This attribute enables or disables the activation of the event channel specified by the AN attribute of the object.

Data type:	Integer
Value:	0 Event channel activation disabled
	1 Event channel activation enabled
Default value:	0
Access:	No restrictions
Object types:	All

AF Action at First Update

This attribute specifies whether the event channel is activated at the first update of the process object or not, see [Section 5.2.8](#) for details.

Data type:	Integer
Value:	0 No activation at first update
	1 Activation also at first update
Default value:	0
Access:	No restrictions
Object types:	All

AH Action on History

This attribute specifies whether updates marked HISTORY (see the CT attribute in [Section 5.3.7](#)) activate the event channel or not.

Data type:	Integer
Value:	0 No event channel activation at HISTORY events
	1 HISTORY events activate the event channel according to the same activation criterion as the real time data
Default:	0
Access:	No restrictions
Object types:	All

AN Action Name

The name of the event channel connected to the process object. Several process objects can have the same event channel. However, each process object can be connected to only one event channel.

Data type:	Text
Value:	Object name
Access:	No restrictions
Object types:	All

EE Event Enabled

This attribute enables or disables the activation of the event object.

Data type:	Integer	
Value:	0	No event object activation
	1	Event object activation
Default value:	0	
Access:	No restrictions	
Object types:	All	

TH Threshold

Event channel activation with threshold means that the event channel connected to the process object is not activated immediately when the value is updated in the process database. Instead, a certain type of algorithm is used to calculate when the activation is to happen.

The purpose of thresholding is to lower the load of frequent event channel activations, for example in cases where the event channel is used to send the value to another control station via a slow communication channel. Event channel activation with threshold applies to analog input process objects. TH attribute implements the threshold.

The algorithm used is an integrating threshold algorithm that works as follows:

1. When AI is updated and the new value differs from the last reported value (the value of AI at the time of last activation), the threshold integral is set to zero and threshold calculation is started.
2. On each calculation cycle (100 ms in current implementation) the time integral of the difference between AI and the reported value is added to the threshold integral. For example, if the reported value is 240.0 and AI is 243.0, value 0.3 is added to the integral.
3. If the absolute value of the integral reaches or exceeds the TH attribute value, the associated event channel is activated and the threshold calculation is stopped.

The algorithm guarantees that if the value is changed once and then stays in that new value, the change is sooner or later reported. The attributes AE and XB are honoured.

AA should be set to UPDATE or NEW VALUE. AF should usually be set to 1 (as it is normally not sensible to filter out the first activation). If AF = 1, the event channel is activated immediately without threshold calculation when the object is updated first time after the start-up of the application.

If AI crosses a warning or alarm boundary or the status of the object (attribute OS) changes, the event channel is activated immediately (or after the delay specified by AD) and the threshold calculation is stopped.

In an HSB system, the intermediate values of the threshold integral are not shadowed due to excessive load. Consequently, the threshold calculation is started from the beginning after a switch-over.

Assigning a new value to TH does not restart the possible on-going threshold calculation. The algorithm always uses the current value of TH.

Data type:	Real
Value:	Non-negative value
Default:	0.0 (no thresholding)
Access:	No restrictions
Object types:	AI

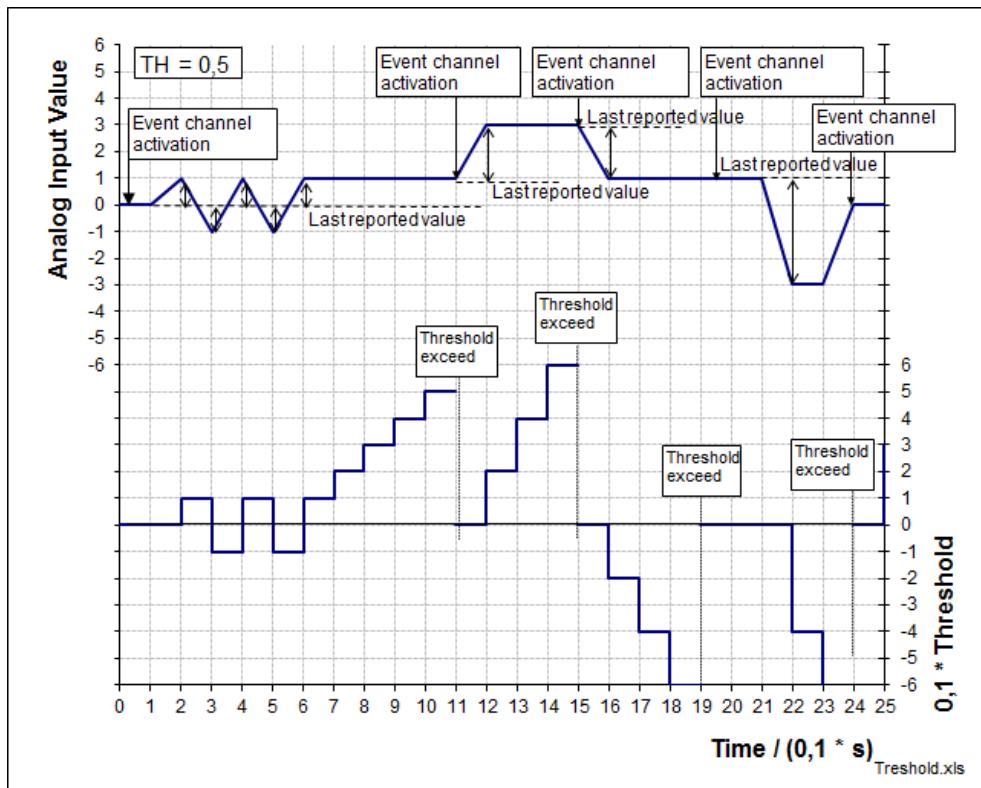


Figure 7: Event channel activation with threshold. Primary (left) y-axis represents the analog input value, here without unit. Secondary (right) y-axis is the time integral of the analog input value. The value of the TH attribute in this example is 0,5. When the value of the threshold reaches or exceeds the value of 0,5, an event channel is activated and the threshold calculation is reset.

5.2.10 Logging of event history

5.2.10.1 General

There are two ways to store event history:

- Using history database (recommended)
- Using event log and history buffer (supported for compatibility)

Application attribute APL:BHP specifies which one is in use. For more information about the HP attribute, see the System Objects manual. The System Configuration manual also explains how to configure storing the event history.

In addition to the common causes of post-processing (attributes listed in [Section 5.2.8](#)), changes of the following attributes are logged:

- Operational state attributes IU (In Use) and SS (Switch State).
- Alarm attributes AC (Alarm Class), AG (Alarm Generation), AR (Alarm Receipt) and LA (Alarm Activation).
- Alarm and warning limits HI (High Input), HW (High Warning), LW (Low Warning), LI (Low Input), HO (High Output) and LO (Low Output).
- Blocking attributes AB (Alarm Blocking), HB (History Blocking), PB (Printout Blocking), UB (Update Blocking) and XB (Action Blocking).

The history logging described in this section can be temporarily blocked by means of the Blocking Attributes (attribute HB, see [Section 5.3](#)).

5.2.10.2 History database

History database consists of history database files that each contain the events for one day. The files are named according to the date as APL_yymmdd.PHD. For example file APL_980115.PHD contains the events logged on January 15th, 1998. For fast access in time stamp order, there is also an index file corresponding each data file. The name extension of the index file is PHI.

Each event in the database contains the values of all attributes of the process object at the time of event. In addition, some pseudo attributes that describe the event itself are included, see [Section 5.3.13](#).

A specialised SCIL function, HISTORY_DATABASE_MANAGER, is designed to handle the history database, see the Programming Language SCIL manual.

The history database is the basis for event lists made by LIB 500 version 4.0.2. and later.

5.2.10.3 History buffer and log

The event history for process objects can also be stored in a history buffer located in the primary memory and in a history log on disk.

The history buffer is implemented as a circular buffer in the global memory pool of SYS600. The size of the buffer is specified by the application attribute APL:BHB. The values of the following attributes at the time of event are stored in the buffer: LN, IX, PT, OX, OV, OS, AL, AS, AR, AT, AM, RT, RM, YT, YM, OF, OI, BL, SB, OR, RA, RB, CT, PB, XB, HB, UB, AB, RI, RX, SE, SP, AZ and CA (Changed Attribute, see [Section 5.3.13](#)). The history buffer is read by process queries (SCIL command #INIT_QUERY and SCIL function PROD_QUERY).

The history log is an ASCII text file. Each event is described as a long text line (377 characters), which contains the values of the following attributes:

GT (3), PT (3), LN (10), IX (3), CA (2), OV (12), OS (5), UN (3), OA (5), OB (2), OT (1), RM (3), RT (10), IU (1), SS (1), HA (1), AB (1), AC (1), AL (1), AR (1), AS (2), AD (3), RC (1), PF (10), OX (30), ST (10), AZ (1), HI (12), HO (12), HW (12), LI (12), LO (12), LW (12), SZ (1), LA (2), NV (1), AG (1), BC (2), CE (1), CL (10), CO (1), CV (10), OF (1), SE (1), SP (1), OI (30), BL (1), SB (1), OR (1), RA (11), RB (11), CT (1), RI (11), RX (10), PH (11), AH (1), HH (1), PB (1), XB (1), HB (1), UB (1), ZE(1), ZD (12), TH (12), OG (5), QL (5), TY (5), IR(1), SU(1), PO(1).

The numbers within parentheses indicate the field length reserved for the attribute value. There is no space between the fields. The attribute names are not included in the log.

If a process object has history buffering (HE = 1), the data of the process object will be copied to the history buffer when any of the following attributes is changed:

- Object value: OV. The OV attribute causes history registrations according to the HA attribute.
- Operational state: SS, IU
- Limit values: HI, LI, LW, HW, HO, LO

- Alarm definition: AC
- Blocking attributes: AB, HB, PB, UB, XB
- Alarm state: AL, AR
- SE, SP. If HA is other than 0, not only a change but also an updating of these attributes causes registration in the history buffer. If HA = 0, these attributes cause no registration in the history buffer.

In addition, each user-defined attribute can be defined to cause a registration in the history buffer.

The history buffer and the history log are the basis for event lists made by LIB 500 version 4.0.1 and earlier versions of application libraries.

5.2.10.4 History configuration attributes

HA History Activation

This attribute specifies the value changes that activate the history logging. For details of the activation criteria, see [Section 5.2.8](#).

Data type:	Integer
Value:	0 ... 15, the activation criterion, see Section 5.2.8
Default value:	0
Access:	No restrictions
Object types:	All

HE History Enabled

This attribute specifies whether logging of event history is enabled for the object.

Data type:	Integer	
Value:	0	History logging is disabled
	1	History logging is enabled
Default value:	0	
Access:	No restrictions	
Object types:	All	

HF History at First Update

This attribute specifies whether history logging is done at the first update of the process object or not, see [Section 5.2.8](#) for details.

Data type:	Integer	
Value:	0	No registration at first update
	1	Registration also at first update
Default value:	0	
Access:	No restrictions	
Object types:	All	

HH History on History

This attribute specifies whether the updates marked HISTORY (see the CT attribute in [Section 5.3](#)) are logged or not.

Data type:	Integer	
Value:	0	HISTORY events are not logged
	1	HISTORY events are logged according to the same activation criterion as the real time data
Default:	0	
Access:	No restrictions	
Object types:	All	

HL History Log Numbers

This attribute is relevant only when the history log and history buffer are used for storing the event history. It specifies the numbers of the printers used as event log printers.

The printers should be configured as virtual printers (without physical correspondence) with printer logging (see the System Configuration manual).

Data type:	Integer	
Value:	A set of logical printer numbers, 1 ... 15, given as bit mask	
	0 = No history log	
Default value:	0	
Access:	No restrictions	
Object types:	All	

5.2.11 Historian logging

The Historian logging attributes specify how the changes of the OV (Object Value) attribute are stored in Historian database(s). For more information about Historian logging, see [Section 12](#).

5.2.11.1 GN Logging Name

This attribute tells the names of Historian database tags that receive data from this process object.

Data type:	Vector	
Element type:	List	
Element value:	DB	The name of the DATABASE type logging profile object that defines the Historian database
	GN	The name of the tag in this Historian database (see below)
Default:	Empty vector	
Access:	Read-only	
Object types:	BI, BO, DB, DI, DO, AI, AO and PC	

The tags are finally named by the Historian database itself. When the tag is not yet created, the GN attribute returns the proposed tag name in square brackets.

5.2.11.2 GP Logging Profile

This attribute specifies the logging profile to be used when logging values to the SYS600 Historian database(s).

Data type:	Text
Value:	Name of an OBJECT type logging profile object. If empty, no logging is done.
Default value:	""
Access:	No restrictions
Object types:	BI, BO, DB, DI, DO, AI, AO and PC

5.2.12 OPC event generation

5.2.12.1 General

OPC events are generated by the SYS600 OPC A&E Server (OAES), which is an implementation of the OPC Alarms & Events Custom Interface Standard Version 1.10 specification by OPC Foundation.

Each SYS600 application that wants to expose its events via the OPC Alarms and Events interface has its own OAES instance (process).

The server is started and stopped by an application attribute (APL:BOE). Starting a server requires a license.

One OAES instance may serve any number of A&E clients.

5.2.12.2 Functionality

The OPC A&E events are tightly coupled with the event and alarm handling of SYS600.

Any process object that generates SYS600 events (loggings to the history database), may act as an OPC A&E event source. History logging and OPC A&E event generation go hand in hand: The process object attributes HE (History Enabled), HA (History Activation) and HF (History at First Update) apply to A&E events as well.

Condition events are associated with SYS600 alarms. An A&E condition is active while the process object is alarming (AL attribute equals 1). Acknowledging the SYS600 alarm (setting AR to 1) acknowledges the A&E condition as well, and vice versa. Alarm blocking blocks the condition events as well.

In addition to condition events, any process object may generate either a simple (input objects) or a tracking event (output objects).

The OV related A&E events to be generated are defined by the associated event handling object of a process object. Each event handling object may define one simple/tracking event and one condition event. All the OPC specified properties of events, conditions and categories are described in event handling objects.

There are two types of predefined event handling objects in SYS600:

- Predefined OV (Object Value) related event handling objects are applied, when no event handling object is associated with the process object (EH = ""). These event handling objects are named according to the process object type and subtype. Some examples are SYS_DI (event handling for Digital Input objects) and SYS_BI_AGO (event handling for Binary Input objects with Alarm Generation from value 0).
- There is a predefined event handling object for each non-OV process object attribute, whose changes are logged in the history database. These are applied regardless of the EH attribute. These event handling objects are named as SYS_aa, where "aa" is the name of the changed attribute. Examples: SYS_AC, SYS_UB.

A&E functionality may be added to the predefined event handling by creating new application default event handling objects. They must be named according to a scheme used for predefined event handling objects, but the prefix SYS is replaced by APL. For example, to define the default A&E event handling for all Binary Input objects with Alarm Generation from value 0, the next steps could be taken:

1. Copy the predefined H-object SYS_BI_AGO to APL_BI_AGO.
2. Add the A&E definitions to APL_BI_AGO.

Now every AG=0 Binary Input object with EH="" follows this new event handling object.

The event message texts of A&E event notifications are identical to SYS600 event texts. Consequently, if they are translated into several languages by the application, an OPC A&E client may choose the language it wants to receive.

5.2.12.3 Event sources

The OPC event source name is defined by the process object attribute ES (Event Source). It is an application-given text, up to 255 characters long.

The event source names are case-sensitive. An OPC event is identified by the source name and the event handling object of the process object. Consequently, two or more process objects may share an event source name (ES), provided that they are connected to different event handling objects (EH). If the ES attribute is left empty, the process object does not generate OPC events.

Normally, hierarchical event source names are used. By default, the following characters are considered as delimiters: dot (.), slash (/), backslash (\) and colon (:). The delimiter character set may be changed by defining the DE sub-attribute of the application attribute APL:BOP, see the System Objects manual.

An example of a valid hierarchical event source name is:

Eastern Substation:Bay 1/Breaker.Position

where three different delimiters and embedded blanks are used. The area browser of OAES recognizes the hierarchy. Source names and area names, such as:

Eastern Substation:Bay 1

may be used as filters in an event subscription.

A process object may generate events of two event types. Any alarming process object may generate condition events. In addition, input objects may generate simple events and output objects may generate tracking events. The properties of the events are defined by the event handling object of the process object, see [Section 6](#).

5.2.12.4 Event categories

The event categories are defined in the event handling objects, see [Section 6](#).

The category id may be any non-zero integer, the category description may be any text up to 255 characters long.

5.2.12.5 Simple and tracking events

The properties of simple and tracking events are described in the event handling objects, see [Section 6](#).

5.2.12.6 Conditions

The properties of conditions, such as condition name, subconditions, severities etc. are described in the event handling object, see [Section 6](#).

Multiple state conditions are supported. Because SYS600 implements multiple state alarms only in case of analog objects, they are applicable only for analog input objects. The subcondition is selected according to the AZ (Alarm Zone) attribute.

As mentioned earlier, SYS600 alarms and A&E conditions go hand in hand:

- Acknowledging a SYS600 alarm by setting AR to 1 also acknowledges the corresponding A&E condition, if any.
- When an A&E client acknowledges a condition, the corresponding SYS600 alarm is acknowledged as well.
- If the SYS600 alarm requires an ack, also the A&E condition requires an ack.

No alarm history is maintained for OPC A&E purposes, only the latest condition is acknowledged.

5.2.12.7 Vendor specific event attributes

One vendor specific event attribute is defined for all event categories. Its attribute description is "CV" (current value), and its attribute id is 1. It contains the value of the OV (Object Value) attribute of the event source process object. Consequently, its value type depends of the process object type.

5.2.12.8 Registration and configuration

The configuration data required by each OAES instance is stored in the application file APL_OAES_aplname.INI, where aplname stands for the name of the SYS600 application. The placement guarantees that the configuration is identical in applications that make up a hot-stand-by pair.

The name of the application is (redundantly) added to the file name to prevent the file being used in an application that is built by copying first an existing application as a template for the new one.

Only one configuration parameter is currently defined:

- CLSID is the unique COM class id of the OPCS instance

An example of the contents of a configuration file might be:

```
CLSID={40838188-4B1F-4C10-A08C-4BD3AF66BBC7}
```

OAES is a self-registering DCOM server. When it starts up, it reads the configuration file and registers itself. If the file does not exist or does not contain the CLSID parameter, it creates a new unique CLSID to itself, adds the CLSID parameter to the configuration file and registers itself.

5.2.12.9 Engineering

Step-by-step procedure for engineering the process database for an A&E server:

1. Design the event categories and their properties: id, description and event type.
2. Design a grouping of process objects according to A&E events they should generate. Each process object of a group sends identical events (apart from the source name) and has

- the same condition (if any). Design the event and condition properties for each group (event handling attributes OS and OC).
3. Create an event handling object for each group or extend the existing event handling object with OPC A&E attributes OG, OS and OC.
 4. If appropriate, create application default event handling objects for various process object types and/or attributes.
 5. Design the event source name hierarchy to be used. If the default delimiters are no good, configure the APL:BOP attribute to define the delimiter characters.
 6. Configure the ES (Event Source) attributes of process objects.
 7. Connect each event source to its event handling object. This is done by setting the EH attribute of the process object.
 8. Set the OE attribute of the application (APL:BOE) to 1. This starts the OAES process.

5.2.13 Printout handling

The attributes in this section specify the printouts related to the process objects, the printing devices and the automatic activation of printing procedures on the occurrence of certain process events. The automatic printing described in this section can be temporarily blocked by means of the Blocking Attributes (attribute PB, see [Section 5.3](#)).

5.2.13.1 LD Listing Devices

The printers to be used for automatic printing of the physical format picture (defined by the PF attribute, see below). The printers are given as a set of logical printer numbers in the range 1 ... 15. The logical printer numbers are determined with the printer mapping attribute (APLn:BPR, see the System Objects manual).

Data type:	Integer
Value:	0 ... 65 534, even numbers Printer numbers given as a bit mask.
Default value:	0
Access:	No restrictions
Object types:	All

Example:

The printer numbers 1, 3 and 5 are alarm and event printers (ABC:PLD2 == 42):

```
#SET ABC:PLD2 = BIT_MASK(1,3,5)
```



In MicroSCADA revisions prior to 8.4.5 SP2 and 9.1, audio-visual alarms were generated only when at least one listing device was connected to the process object (LD > 0). In later revisions, audio-visual alarms are independent of the LD attribute.

5.2.13.2 PA Printout Activation

This attribute specifies the value changes that activate the printout. For details of the activation criteria, see [Section 5.2.8](#).

Data type:	Integer
Value:	0 ... 15, the activation criterion, see Section 5.2.8
Default value:	0
Access:	No restrictions
Object types:	All

5.2.13.3 PF Physical Format

The name of the automatically printed picture. This picture is also printed with the command #LIST (see [Section 4](#)). Each object can have its own physical format picture, or several objects can share the same picture. When an automatic printout is activated, a set of argument variables, so called snapshot variables, are passed to the format picture, see [Section 5.2.8](#).

Data type:	Text
Value:	Picture name
Access:	No restrictions
Object types:	All

5.2.13.4 PH Printout on History

The attribute specifies whether updates marked HISTORY (see the CT attribute [Section 5.3.7](#)) activate automatic printing or not.

Data type:	Integer	
Value:	0	No printout activation at HISTORY events
	1	HISTORY events are printed according to the same activation criterion as the real time data
Default:	0	
Access:	No restrictions	
Object types:	All	

5.2.13.5 PU Printout at First Update

This attribute specifies whether the printout is activated at the first update of the process object or not, see [Section 5.2.8](#) for details. This attribute setting applies to alarm picture signalling as well as audio alarms.

Data type:	Integer	
Value:	0	No activation at first update
	1	Activation also at first update
Default value:	0	
Access:	No restrictions	
Object types:	All	

5.2.14 Network topology

The attributes described below are not configurable by SCIL, they are configured automatically when a network topology model is imported into the process database.

5.2.14.1 CN Connected Network Objects

The network topology (NT) objects connected to this NT object.

Data type:	Vector
Element type:	List
Element value:	LN Logical name of a connected object IX Logical index of a connected object
Default value:	Empty vector
Access:	Read-only
Object types:	NT

5.2.14.2 NM Network Topology Model

The network topology model of the object.

Data type:	Text
Value:	The name of the topology model the object belongs to
Default value:	Empty text
Access:	Read-only
Object types:	NT

5.2.14.3 NO Network Object Connection

Reference to the connected network topology (NT) object.

Data type:	List
Value:	Attributes specifying the referenced object:
	LN Logical name IX Logical index
Default:	Empty list
Access:	Read-only
Object types:	All

When the value of the process object changes, recalculation of the network topology is triggered via this attribute.

5.2.14.4 NS Network Object Subtype

The device type of a network topology object.

The possible types depend on the topology schema. The values listed below correspond to the POWER schema.

Data type:	Integer	
Value:	0	None
	1	Earth
	2	Power

Table continues on next page

3	Busbar
4	Switching device
5	Transformer winding 1
6	Transformer winding 2
7	Transformer winding 3
8	Transformer winding 4
9	Line indicator
10	Truck
11	Generator
Default value:	0
Access:	Read-only
Object types:	NT

5.2.14.5 RO Referenced Objects

The (non-NT) process objects referenced by this object.

Data type:	Vector
Element type:	List
The name of the attribute tells the type of the reference, the value specifies the referenced object. The reference types of the POWER schema are the following:	
TS	Topological state indicator
LV	Voltage level indicator
CH	The child switching device of a truck
AP	The auxiliary plug of a truck
PR	The truck object of the child
Element value:	LN Logical name of the referenced object
	IX Logical index of the referenced object
Default value:	Empty vector
Access:	Read-only
Object types:	NT

5.2.15 Miscellaneous attributes

5.2.15.1 Counter definition attributes

CE Counter Enabled

Operation counting is taken into use by setting this attribute to one.

Data type:	Integer	
Value:	0	No counting
	1	Counting in use
Default:	0	
Access:	No restrictions	
Object types:	BI, BO, DB and OE	

CL Counter Limit

The upper limit for the counter. When the counter value (the CV attribute) exceeds this limit, counter overflow (the CO attribute) is set.

Data type:	Integer
Default:	0
Access:	No restrictions
Object types:	BI, BO, DB and OE

5.2.15.2 SCIL application attributes

The following attributes are reserved for various applications to be used in SCIL programs. They have no functionality within the base system.

DX Directive Text

This attribute is reserved by Hitachi Power Grids and should not be used in application programs.

FI Free Integer

An integer attribute that can be freely used for any application purpose.

Data type:	Integer (32 bits)
Default value:	0
Access:	No restrictions
Object types:	All

FX Free Text

A text attribute that can be freely used for any application purpose.

Data type:	Text
Value:	Up to 63 Unicode characters
Default value:	""
Access:	No restrictions
Object types:	All

GI Gateway Information

An integer attribute reserved for use of standard gateway applications, such as COM500i.

Data type:	Integer
Default value:	0
Access:	No restrictions.
Object types:	All

RI Reserved Integer

An integer attribute reserved for use of standard application software, such as LIB 500.

Data type:	Integer
Default value:	0
Access:	No restrictions.
Object types:	All

RX Reserved Text

A text attribute reserved for use of standard application software, such as LIB 500.

Data type:	Text
Value:	Up to 63 characters
Default value:	""
Access:	No restrictions.
Object types:	All

RZ Reserved Z

An integer attribute reserved for use of tagout functionality.

Data type:	Integer
Default value:	0
Access:	No restrictions.
Object types:	All

5.3 Dynamic process object attributes

This section describes the process object attributes that reflect the real time state of the process. The attributes are grouped into the following subsections:

Section 5.3.1	Object value: AI, AO, BI, BO, BS, DB, DI, DO, NT, OE, OV, PC, SX, TS
Section 5.3.2	Time and validation stamps: CS, OS, RM, RQ, RT
Section 5.3.3	Alarm and warning states: AL, AM, AQ, AR, AS, AT, AV, AZ, KT, KM, KQ, WQ, YM, YQ, YT
Section 5.3.4	Blocking attributes: AB, HB, PB, XB, UB
Section 5.3.5	Operation counter attributes: CO, CV
Section 5.3.6	Minimum and maximum values: MM, MQ, MT, MV, XM, XQ, XT, XV
Section 5.3.7	Stamps set by the communication system: BL, CT, OR, RA, RB, SB, TM
Section 5.3.8	S.P.I.D.E.R. RTU specific attributes: EP, OF, SE, SP
Section 5.3.9	IEC, REX and DNP specific attributes: OG, QL, TY
Section 5.3.10	OPC Event objects: AK, CK, CM, CN, CQ, ID, NS, OQ, QU, SE, SN, VC, VM, VQ, VT

Table continues on next page

Section 5.3.11	File transfer attributes:	DC, FF, FN, FP, FT, ID, ST
Section 5.3.12	Network topology attributes:	LE, LP, LV, LX, ND, NF
Section 5.3.13	Event history attributes:	CA, ED, EM, EQ, ET, EX, HD, HM, HQ, HT, MX, PV, US

The attributes are in alphabetical order within each sub-section.

5.3.1 Object value

The attributes described in this section represent the actual value of the object. For real objects (process objects connected to the process), the object values of input type are automatically updated from the process stations. The object values of output type are sent out to the process stations when set with the #SET command.

5.3.1.1 AI Analog Input

An analog input value (measured value) from the station via the communication system to the base system. The value is scaled in the process database according to the scale defined by the SN attribute (see [Section 5.2.4](#) and [Section 7](#)).

Data type:	Real, if IR = 0
	Integer, if IR = 1
Default value:	0
Access:	Read, conditional write. The attribute can be written only if the switch state is manual (SS = 1) or fictitious (SS = 3), or if its UN = 0 and SS = 2.
Object types:	AI

5.3.1.2 AO Analog Output

An analog output value (for example a set value, an analog setpoint, general output) from the base system to the station, via the communication system. The value is scaled according to the scale given by the SN attribute (see 3.2.4.) before it leaves the process database.

Data type:	Real, if IR = 0
	Integer, if IR = 1
Default value:	0
Access:	Read, conditional write. The attribute cannot be written if the process object is OFF (SS = 0).
Object types:	AO

5.3.1.3 BI Binary Input

A binary input signal (indication) from the station to the base system via the communication system.

Data type:	Integer
Value:	0 or 1
Default value:	0
Access:	Read, conditional write. The attribute can be written with SCIL if the switch state is manual (SS = 1) or fictive (SS = 3) or if its UN = 0 and SS = 2, but not otherwise.
Object types:	BI

5.3.1.4 BO Binary Output

A binary output signal (control signal, object command, regulation command) from the base system via the communication system to the station.

Data type:	Integer
Value:	0 or 1
Default value:	0
Access:	Read, conditional write. The attribute cannot be written if the process object is OFF (SS = 0).
Object types:	BO

5.3.1.5 BS Bit Stream

An output and input signal of bit string type.

Data type:	Bit string
Value:	Maximum length 8 176 bits, if connected to the process, otherwise 65 535 bits
Access:	Read, conditional write. The attribute cannot be written if the process object is OFF (SS = 0).
Object types:	BS

Bit stream objects may also be used to receive text (VT_BSTR) data from an OPC Data Access Server. In this case, the maximum length of the bit string is 65528 bits, equivalent to 8191 characters. The BS value is converted to text by the SCIL function TYPE_CAST, see the Programming Language SCIL manual.

Example:

```
@MY_TEXT = TYPE_CAST(ABC:PBS1, "TEXT") ;Read the BS value as text
```

5.3.1.6 DB Double Binary Indication

A double binary indication from the station. In some stations, double binary indications are handled as two single binary input values. In SYS600 the first bit means "closed", and the second bit "open", by default.

Data type:	Integer	
Value:	0	Intermediate position
	1	Closed
	2	Open
	3	Faulty position

Table continues on next page

Default value:	0
Access:	Read, conditional write. The attribute can be written only if the switch state is manual (SS = 1) or fictitious (SS = 3) or if its UN = 0 and SS = 2.
Object types:	DB

The semantics of values 1 and 2 may be swapped by STA:BSM (Topological State Mapping) attribute, or application wide by APL:BSM attribute. For details, see the System Objects manual.

5.3.1.7 DI Digital Input

A digital input value from the station via the communication system to the base system.

Data type:	Integer
Value:	0 ... 65 535 (16 bits)
Default value:	0
Access:	Read, conditional write. The attribute can be written only if the switch state is manual (SS = 1) or fictitious (SS = 3) or if its UN = 0 and SS = 2.
Object types:	DI

5.3.1.8 DO Digital Output

A digital output value (digital setpoint) from the base system via the communication system to the station.

Data type:	Integer
Value:	0 ... 65 535 (16 bits)
Default value:	0
Access:	Read, conditional write. The attribute cannot be written if the process object is OFF (SS = 0).
Object types:	DO

5.3.1.9 NT Network Object State

The calculated state of a network topology object.

The possible states of a network topology object depend on the topology schema. The values listed below correspond to the POWER schema.

Data type:	Integer	
Value:	0	Unknown
	1	Unpowered
	2	Uncertain
	3	Powered
	4	Error
	5	Earthed
Default value:	0	
Access:	Read-only	
Object types:	NT	

5.3.1.10 OE OPC Event

A numeric value corresponding to the state of the OPC condition or to the message last received (see [Section 5.3.10](#)).

Data type:	Integer
Value:	0 ... 255
Default value:	0
Access:	Read-only
Object types:	OE

5.3.1.11 OV Object Value

The OV attribute of output objects of IEC/REX type (BO, DO, AO, BS) is set with syntax given in [Table 4](#). For an object of IEC/REX type in auto state only output objects (BO, DO, AO, BS) is set with syntax given in [Table 4](#).

The intended use of the OV attribute is:

1. As a comprehensive name for one of the attributes AI, AO, BI, BO, BS, DI, DO, DB, PC, OE, NT and FT depending on the object type. For user-defined object types, the programmer selects which attribute, if any, will be the OV attribute. Setting the OV attribute of an input object is meaningful only in case the object is in manual or fictitious state.
2. As default attribute when no attribute name is given at assigning a value of an object. For example, the #SET commands:

```
#SET A:PAI1 = 0
#SET A:POV1 = 0
#SET A:P1 = 0
```

of an analog input object are equivalent.

3. For input objects of other type than OE and output objects of other type than IEC, the OV attribute can be set manually together with the attributes listed in the example according to the following model:

```
#SET name:P[OV]i = LIST(OV=exp1[,RT=exp2, RM=exp3, OS=exp4, OF= exp5,
BL= exp6,-SB= exp7, TM= exp8,OR= exp9, RA= exp10, RB= exp11, CT=
exp12,-
TY= exp13, QL= exp14, OG= exp15])
```

where 'exp1', 'exp2', 'exp3' etc. are expressions assigned to the attributes. The part within square brackets can be omitted, whereby the RM and OS attributes are set to 0. Attributes RT and RM are assigned values at registration time if not given in the list. Attributes TY, QL and OG apply to IEC/REX type objects only. For objects of type IEC/REX also the OV attribute is optional.

Also the stamps set in the stations (see [Section 5.3.7](#)) can be written in this way.

Data type:	Integer, real or bit string
Value:	Integer 0 or 1 for binary objects
	Real or integer for analog objects
	Integer for DI, DO, PC, FT, OE and NT objects
	Integer 0 ... 3 for DB objects
	Bit string for BS objects

Table continues on next page

Default value:	0
Access:	See the AI, AO, BI, BO, BS, DI, DO, DB, PC, FT, OE and NT attributes
Object types:	All

Example:

```
#SET A:P2 = LIST(OV=3.0,RT=B:PRT,RM=B:PRM)
```

or

```
#SET A:POV2 = LIST(OV=3.0,RT=B:PRT,RM=B:PRM)
```

5.3.1.12 PC Pulse Counter

A pulse counter value from the station. The type of reading when the process object belongs to a S.P.I.D.E.R. RTU (“End of period” or “intermediate”), is given by the EP attribute of the object, see [Section 5.3.8](#).

Data type:	Integer
Value:	The value range is station type and protocol dependent.
Default value:	0
Access:	Read, conditional write. The attribute can be written only if the switch state is manual (SS = 1), or fictitious (SS = 3) or if its UN = 0 and SS = 2.
Object types:	PC

5.3.1.13 SX State Text

A descriptive text of the object value.

Data type:	Text
Value:	A descriptive text of the object state in the current language
Default value:	Empty text
Access:	Read-only
Object types:	All

The text is obtained from the event handling object referenced by the EH attribute of the process object.

5.3.1.14 TS Topological State

The value of the object (OV) interpreted as a topological state.

Data type:	Integer
Value:	0 .. 3 In the topology schema POWER, the semantics of numeric values is as follows:
0	Intermediate (middle) state
1	Closed
2	Open

Table continues on next page

	3	Faulty state
Access:	Read-only	
Object types:	BI, DB	

For Binary Input (BI) objects, BI value 0 represents topological state 2 (open) and value 1 represents state 1 (closed) in the POWER schema.

For Double Binary (DB) objects, TS is determined according to the Topological State Mapping (SM) attribute of the station object and/or the application. If the SM attributes are not set, TS follows the value of DB (in any schema).

For details of the SM attribute (STAn:BSM and APLn:BSM), see the System Objects manual.

5.3.2 Time and validation stamps

These attributes are related to the object value. Each update of the object value in the process database gets a validation stamp (the OS attribute) and time stamp (the RT, RM and RQ attributes), regardless of whether the object value changes or not. For output objects, the CS attribute stores the success code of the latest control attempt for tracking purposes.

5.3.2.1 CS Control Status

Status of the latest attempt to control the object.

Data type:	Integer
Value:	0 ... 65535 The status code of the latest attempt to set the OV attribute of the object by SCIL.
Default value:	10 (NOT_SAMPLED_STATUS)
Access:	Read-only
Object types:	BO, DO and AO

5.3.2.2 OS Object Status

The status code of the process object. This attribute indicates the reliability of the object value (the OV attribute).

Data type:	Integer
Value:	0, 1, 2, 3 or 10: 0 OK_STATUS 1 FAULTY_VALUE_STATUS. The station has marked the process object value faulty. An update with this status sets the object to alarm state (AL = 1), provided that AC > 0 (Section 4.2). The alarm state prevails until a new update comes with OS = 0. 2 OBSOLETE_STATUS. The value is uncertain, it is possible that it is not up-to-date. The connection to the station is (or has been) lost. At application start-up, non-manual output objects are marked with this status. 3 FAULTY_TIME_STATUS. The station has marked the registration time faulty.

Table continues on next page

10 NOT_SAMPLED_STATUS. The object value has not yet been initialized. At application start-up, non-manual input objects are marked with this status. This status is also set when a modification of an object invalidates its object value, for example SS or IU or the object address is changed.

Access: Read-only. The OS attribute can be set manually along with the OV attribute, see the OV attribute, [Section 5.3.1](#).

Object types: All

5.3.2.3 RM Registration Milliseconds

The milliseconds of the registration time. See attribute RT for the originator of this attribute value.

Data type: Integer

Value: 0 ... 999

Unit: Milliseconds

Access: Read-only. The RM attribute can be set manually along with the RT and OV attributes, see the OV attribute, [Section 5.3.1](#).

Object types: All

5.3.2.4 RQ Registration Qualified Time

The time when the object was last updated as a qualified time. See attribute RT for the originator of this attribute value.

Data type: List

Value: Qualified time, attributes

CL Time

MS Integer, milliseconds

DS Boolean, daylight saving

Initial value: Application start-up time

Access: Read-only. The RQ attribute can be set manually along with the OV attribute, see [Section 5.3.1](#).

Object types: All

5.3.2.5 RT Registration Time

The time when the object was last updated.

The time stamp may originate from the station, NET or the base system. If the update from NET contains a time stamp (stamped by the station or NET itself), the time is copied to the RT attribute. If the time stamp does not contain a date, the date is supplied by the base system. If the update does not have a time stamp at all, the RT attribute is set according to the base system time. When an object is updated with SCIL (with #SET), the time stamp is always given by the base system.

Output objects are time stamped before the command is sent to NET. The RT attribute of BO type objects is also updated when SE or SP is set ([Section 5.3.8](#)).

If an object has an alarm delay (the AD attribute, [Section 4.2](#)), the RT attribute is updated when the alarm delay expires.

Data type:	Time
Initial value:	Application start-up time
Access:	Read-only. The RT attribute can be set manually along with the OV attribute, see Section 5.3.1 .
Object types:	All

5.3.3 Alarm and warning states

The attributes in this section show the alarm and warning states of the process objects. Alarm and warning handling is defined by the alarm handling attributes in [Section 5.2.6](#) and the limit values in [Section 5.2.7](#).

5.3.3.1 AL Alarm

This attribute indicates whether alarm is active or not.

Data type:	Integer	
Value:	0	No alarm
	1	Alarm is active
Access:	Read-only	
Object types:	All	

5.3.3.2 AR Alarm Receipt

Status of acknowledgement. This attribute indicates whether or not the alarm has been acknowledged.

Data type:	Integer	
Value:	0	The alarm is not acknowledged
	1	The alarm is acknowledged or there is no demand for acknowledgement
Access:	No restrictions	
Object types:	All	

5.3.3.3 AS Alarm State

The alarm state is a number calculated from the alarm class (AC), the alarm (AL) and the state of acknowledgement (AR).

Attribute WC is used instead of attribute AC when the alarm is generated by low warning or high warning state.

Data type:	Integer
Value:	0 ... 14, see below
Access:	Read-only
Object types:	All

When an alarm arises, the attribute gets the value of AC (1 ... 7), if RC = 1, and AC + 7, if RC = 0. If the alarm is generated by low warning or high warning, the attribute gets the value of WC (1 ... 7), if WR = 1. When the alarm is acknowledged, 7 is added to the attribute value. When the alarm

disappears, the attribute gets the value 0, supposing that it has been acknowledged, or there is no demand for acknowledgement.

5.3.3.4 AV Alarm Severity

Alarm severity is calculated separately using other configured process object alarm attributes. For analog values the alarm severity value is 1 for warning limit (LW, HW) and 2 for alarm limit (LI, HI). For all other types the alarm severity follows the configured alarm class value (AC). The higher number represents higher severity.

Data type:	Integer
Value:	0...7 0 = No alarm
Default value:	0
Access:	Read-only
Object types:	All

5.3.3.5 AM Alarm Milliseconds

The milliseconds of the alarm time (see the AT attribute below), that is, the time when an alarm last arose or was cleared. Generally, the alarm milliseconds are the same as the RM attribute of the update that caused the change of alarm state. However, in case the object has an alarm delay there may be a difference, see the AT attribute.

Data type:	Integer
Value:	0 ... 999
Initial value:	Application start-up time
Unit:	Milliseconds
Access:	Read-only
Object types:	All

5.3.3.6 AQ Alarm Qualified Time

The alarm time when an alarm last arose or was cleared (as a qualified time). Generally, the alarm time is the same as the RQ attribute of the update that caused the change of alarm state. If the object has an alarm delay, the AQ attribute is updated when an alarm occurs. The RQ attribute is updated when the alarm delay expires.

Data type:	List
Value:	Qualified time, attributes
	CL Time
	MS Integer, milliseconds
	DS Boolean, daylight saving
Initial value:	Application start-up time
Access:	Read-only
Object types:	All

5.3.3.7 AT Alarm Time

The alarm time when an alarm last arose or was cleared (in one second resolution). Generally, the alarm time is the same as the RT attribute of the update that caused the change of alarm

state. If the object has an alarm delay, the AT and AM attributes are updated when an alarm occurs. The RT and RM attributes are updated when the alarm delay expires.

Data type:	Time
Initial value:	Application start-up time
Access:	Read-only
Object types:	All

5.3.3.8 AZ Alarm Zone

The alarm and warning state of the object, see [Figure 8](#). Depending on the SZ attribute, the value of the AZ is set either by SYS600 according to the limit values or by the station, see [Section 5.2.7](#).

Data type:	Integer
Value:	0 ... 4: 0 Normal state 1 Low alarm 2 High alarm 3 Low warning 4 High warning
Access:	Read-only
Object types:	AI and user-defined types with real or integer value

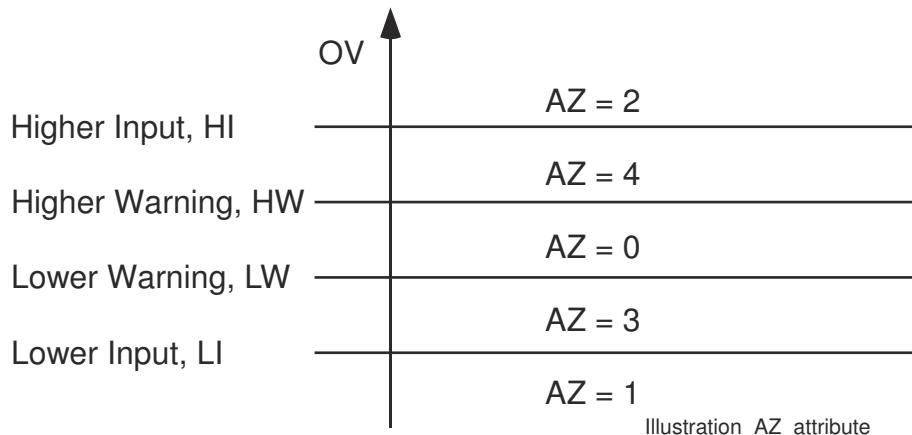


Figure 8: An illustration of the AZ attribute (SZ = 1)

5.3.3.9 KM Acknowledgement Milliseconds

The milliseconds of the alarm acknowledgement time (see the KT attribute below). The value is set to 0 when the alarm is raised.

Data type:	Integer
Value:	0 ... 999
Initial value:	0

Table continues on next page

Unit:	Milliseconds
Access:	Read-only
Object types:	All

5.3.3.10 KQ Acknowledgement Qualified Time

The time when the alarm was acknowledged (as a qualified time). The default value is set when the alarm is raised.

Data type:	List
Value:	Qualified time, attributes
	CL Time
	MS Integer, milliseconds
	DS Boolean, daylight saving
Initial value:	Application start-up time
Access:	Read-only
Object types:	All

5.3.3.11 KT Acknowledgement Time

The time when the alarm was acknowledged (in one second resolution). The default value is set when the alarm is raised.

Data type:	Time
Initial value:	1978-01-01 00:00 (zero time)
Access:	Read-only
Object types:	All

5.3.3.12 WQ Warning Qualified Time

The time when the object entered the warning state (as qualified time).

Data type:	List
Value:	Qualified time, attributes
	CL Time
	MS Integer, milliseconds
	DS Boolean, daylight saving
Initial value:	Application start-up time
Access:	Read-only
Object types:	AI

5.3.3.13 YM Alarm On Time Milliseconds

The milliseconds of the time when an alarm last occurred in the object. This attribute is identical to AM while the object alarm is active, but unlike the AM attribute, the YM attribute is not updated when the alarm is cleared.

Data type:	Integer
Value:	0 ... 999
Unit:	Milliseconds
Initial value:	0
Access:	Read-only
Object types:	All

5.3.3.14 YQ Alarm On Qualified Time

The time when an alarm last occurred in the object (as a qualified time). This attribute is identical to the AQ attribute while the object alarm is active. Unlike the AQ attribute, the YQ attribute is not updated when the alarm is cleared.

Data type:	List
Value:	Qualified time, attributes
	CL Time
	MS Integer, milliseconds
	DS Boolean, daylight saving
Initial value:	CL = 0, MS = 0, DS = FALSE
Access:	Read-only
Object types:	All

5.3.3.15 YT Alarm On Time

The time when an alarm last occurred in the object (in one second resolution). This attribute is identical to the AT attribute while the object alarm is active. Unlike the AT attribute, the YT attribute is not updated when the alarm is cleared.

Data type:	Time
Initial value:	0
Access:	Read-only
Object types:	All

5.3.4 Blocking attributes

The following attributes are used to temporarily block the updating of a process object or to block alarm generation and post-processing (printouts, event channel activation and history logging) normally caused by process object update. The values of the blocking attributes are stored on disk. Like the dynamic attributes, they are included in the Snapshot variable list. The setting of any of the attributes is logged in the history database. Also, an event is generated.

5.3.4.1 AB Alarm Blocking

Blocks alarm generation of the object.

Data type:	Integer
Value:	0 No blocking
	1 Alarm generation is blocked

Table continues on next page

Default value:	0
Access:	No restrictions
Object types:	All

When AB is set to 1, the alarm state of the object is cleared. When it is reset to 0, the alarm state is recalculated unless prevented by revision compatibility (APL:BRC) setting DONT_RECALCULATE_AL_AFTER_ALARM_BLOCKING. No printout or event channel is activated even if the alarm state is changed.

5.3.4.2 HB History Blocking

Blocks logging of the history of the object.

Data type:	Integer	
Value:	0	No blocking
	1	History logging is blocked
Default:	0	
Access:	No restrictions	
Object types:	All	

When history registration is blocked, no events are written into the history database. Consequently, no events of the object are seen in the event list

5.3.4.3 PB Printout Blocking

Blocks printout generation of the object.

Data type:	Integer	
Value:	0	No blocking
	1	Printouts are blocked
Default:	0	
Access:	No restrictions	
Object types:	All	

5.3.4.4 UB Update Blocking

Blocks updating of the value of the object (OV attribute).

Data type:	Integer	
Value:	0	No blocking
	1	Updating is blocked
Default:	0	
Access:	No restrictions	
Object types:	All	

When the object is blocked (UB = 1), it is not updated by the process, nor from SCIL. An error is raised if the OV value is tried to be set by SCIL. Consequently, both input and output via the OV attribute is blocked. The OV attribute may, however, be read by SCIL.

When UB is set to 1, the OS attribute of the object is set to OBSOLETE_STATUS (2) to indicate that the value may be outdated.

5.3.4.5 XB Activation Blocking

Blocks event channel activation.

Data type:	Integer	
Value:	0	No blocking
	1	Event channel activation is blocked
Default:	0	
Access:	No restrictions	
Object types:	All	

5.3.5 Operation counter attributes

These attributes are used for counting the operations of an object. They can be used, for example, for monitoring the need for service. The attributes are valid for BI, BO, DB and OE objects.

5.3.5.1 CO Counter Overflow

This attribute is set to 1 when the CL attribute (Counter Limit) is exceeded.

Data type:	Integer	
Value:	0	No overflow
	1	Overflow
Access:	Read-only	
Object types:	BI, BO, DB and OE	

The attribute is automatically reset when the CV attribute is reset.

5.3.5.2 CV Counter Value

This attribute counts the changes of the object value (see below).

Data type:	Integer	
Initial value:	0	
Access:	No restrictions	
Object types:	BI, BO, DB and OE	

For BI, BO and DB objects, the counter is incremented when the object value changes to 1 (from any other value).

For OE objects, the counter is incremented when the condition enters the active state (CONDITION events) or on every event (SIMPLE and TRACKING events).

The attribute can be reset with SCIL (no automatic reset). The value is stored on disk.

5.3.6 Minimum and maximum values

The attributes in this section apply to analog input (AI) objects. All the minimum and maximum attributes are stored on RAM only, not on disk. They are passed as snapshot variables to format pictures and event channels.

5.3.6.1 MM Minimum Time Milliseconds

The milliseconds of the time when the minimum value (the MV attribute) occurred.

Data type:	Integer
Value:	0 ... 999
Unit:	Milliseconds
Access:	Read-only
Object types:	AI

5.3.6.2 MQ Minimum Qualified Time

The time (as qualified time) when the minimum value (the MV attribute) occurred.

Data type:	List
Value:	Qualified time, attributes
	CL Time
	MS Integer, milliseconds
	DS Boolean, daylight saving
Access:	Read-only
Object types:	AI

5.3.6.3 MT Minimum Time

The time in seconds when the minimum value (the MV attribute) occurred.

Data type:	Time
Access:	Read-only
Object types:	AI

5.3.6.4 MV Minimum Value

Records the lowest value of the AI attribute since the last reset. At the first update of the object after application start-up, the MV attribute is reset to the current value. It may also be reset by SCIL by writing any value into the MV attribute (normally it is reset to the current value of AI).

Data type:	Real, if IR = 0
	Integer, if IR=1
Default:	The value of the object at application start-up
Access:	No restrictions
Object types:	AI

5.3.6.5 XM Maximum Time Milliseconds

Milliseconds of the time when the maximum value (the XV attribute) occurred.

Data type:	Integer
Value:	0 ... 999
Unit:	Milliseconds
Access:	Read-only
Object types:	AI

5.3.6.6 XQ Maximum Qualified Time

The time (as qualified time) when the maximum value (the XV attribute) occurred.

Data type:	List
Value:	Qualified time, attributes
	CL Time
	MS Integer, milliseconds
	DS Boolean, daylight saving
Access:	Read-only
Object types:	AI

5.3.6.7 XT Maximum Time

The time in seconds when the maximum value (the XV attribute) occurred.

Data type:	Time
Access:	Read-only
Object types:	AI

5.3.6.8 XV Maximum Value

Records the highest value of the AI attribute since last reset. At the first update of the object after application start-up, the XV attribute is reset to the current value. It may also be reset by SCIL by writing any value to the XV attribute (normally it is reset to the current value of AI).

Data type:	Real, if IR = 0
	Integer, if IR=1
Default:	The value of the object at application start-up
Access:	No restrictions
Object types:	AI

5.3.7 Stamps set by the communication system

The attributes in this section contain information set in NET on the basis of stamps set by the process units (in the relays or RTUs) in accordance with the used protocol. If the communication protocol supports these attributes, they are updated in the process database along with the object value. The following rules apply for the BL, SB, TM, OR and OF attributes:

- A change of a quality attribute generates an event if EE = 1.
- A change of a quality attribute activates an event channel, a printout and/or history logging if the activation is enabled (AE == 1, LD >> 0 or HE == 1) and the activation criterion (AA, PA or HA) is NEW VALUE or UPDATE.
- In such activation, the changed attribute is reported as the value of CA pseudo attribute. If more than one attribute is changed at the same time, each change will be reported separately in any order. For example, if OV changes from 0 to 1 and SB from 1 to 0, two activations occur, one with CA == "BI", BI == 1 and SB == 0, the other with CA == "SB", BI == 1 and SB == 0.
- When the switch state (SS) or the substitution state (SU) of the object is changed, the quality attributes are set to 0.

RA and RB attributes are of informative character and do not affect the function of the process object. The CT attribute affects the activation of the post-processing.

These attributes are supported by multiple protocols, see protocol specific manuals for details. If a protocol does not support the attributes, they can still be used, but must be set with SCIL. However, the attributes cannot be set one by one using the #SET command. They may only be set along with the OV attribute using list value in the #SET command, see the OV attribute in [Section 5.3.1](#).

All attributes in this section are stored only in RAM.

5.3.7.1 BL Blocked

The updating of the value has been blocked in the relay.

Data type:	Integer	
Value:	0	Not blocked
	1	Blocked
Access:	Read-only. Can be written together with the OV attribute, see Section 5.3.1 .	
Object types:	All	

5.3.7.2 CT Cause of Transmission

The type of the data transmission from the station to the process database. The function of the CT attribute can be determined by setting a base system STY object attribute, see the System Objects manual.

There are the following types of data transmission:

UNKNOWN	Updating is not marked regarding the type of transmission because the protocol does not support the attribute
SPONTANEOUS	Spontaneous updating caused by a change in the station
INTERROGATED	The transmission started on an interrogation from SYS600
HISTORY	The transmission contains history data from the history buffer of the station

If CT = UNKNOWN, the process object is updated in the process database and the consequential actions (post-processing) takes place according to the activation attributes, that is, the update is regarded as spontaneous.

If CT = SPONTANEOUS, there is a slight difference in the case of the first update of the process object (when OS is 10 before the event). Post-processing (printout, event channel activation and history logging) is done regardless of the value of the corresponding control attribute (PU, AF or HF, [Section 5.2](#)).

If CT = INTERROGATED and the object value has not changed, there is a slight difference in the post-processing when the activation criteria is UPDATE, see the AA, HA and PA attributes in [Section 5.2](#).

If CT = HISTORY, the process database is not updated at all. Only post-processing (printout, event channel activation and/or history logging) is done. The snapshot variable list and the history buffer (and history log) entry contains the received OV value and the received communication protocol attribute values. The rest of attributes are copied from the process object. The pseudo-attribute CA is set to "CT".

Data type:	Integer
Value:	0 ... 255 as defined by the CT attribute of the station type (the STYn:BCT attribute, see the System Objects manual. The following values apply to stations of type REX:
0	UNKNOWN (applies to all station types)
1	SPONTANEOUS
2	INTERROGATED
3	HISTORY
	Value 0 may mean that the attribute is not supported by the protocol.
Default value:	0
Access:	Read-only. The attribute can be written along with the OV attribute, see Section 5.3.1 .
Object types:	All

5.3.7.3 OR Out of Range

The station cannot handle the signal read from the process. An example could be that the signal value is larger or smaller than the range supported by an analog/digital converter.

An alarm is generated when this attribute is set to 1 unless prevented by revision compatibility (APL:BCR) value "NO_ALARM_BY_OR_AND_OF".

Data type:	Integer
Value:	0 OK
	1 Out of range
Access:	Read only. The attribute can be written along with the OV attribute, see Section 5.3.1 .
Object types:	All

5.3.7.4 RA Reserved A

RB Reserved B

These attributes are used for protocol dependent data in various ways in different protocols. In slave protocols, these attributes are used by the COM500i application. See the protocol specific manuals for more information.

In the RP571 protocol these attributes are used as follows:

RA	Relative time
RB	Event number
Data type:	Integer
Access:	Read only. The attribute can be written along with the OV attribute, see Section 5.3.1 .
Object types:	All

5.3.7.5 SB Substituted

The value read from the process has been substituted by another value in the station. For example, the value has been changed manually on the relay front panel.

Data type:	Integer	
Value:	0	Not substituted
	1	Substituted
Access:	Read-only. The attribute can be written along with the OV attribute, see Section 5.3.1 .	
Object types:	All	

5.3.7.6 TM Test Mode

The device is in test mode.

Data type:	Integer	
Value:	0	Not in test mode
	1	In test mode
Access:	Read-only. The attribute can be written along with the OV attribute, see Section 5.3.1 .	
Object types:	All	

5.3.8 S.P.I.D.E.R. RTU specific attributes

5.3.8.1 EP End of Period

This attribute tells the type of pulse counter reading. Each pulse counter update has this attribute.

Data type:	Integer	
Value:	0	Intermediate reading
	1	End of period reading
Initial value:	0	
Access:	Read-only	
Object types:	PC	

5.3.8.2 OF Overflow

This attribute indicates whether there is an overflow in the event recording buffer or pulse counter history buffer of the RTU.

The attribute is only valid for pulse counters and event recording objects (see 3.3). Each update of these objects, originating from the RTU, contains the OF attribute.

An alarm is generated when this attribute is set to 1, unless prevented by revision compatibility (APL:BRC) value "NO_ALARM_BY_OR_AND_OF".

Data type:	Integer
Values:	0 No overflow 1 Overflow
Access:	Read-only
Object types:	All (meaningful only for some types, however)

5.3.8.3 SE Selection

Selecting the object means that SYS600 performs a check-back before executing the operation.

Selection of a IEC or REX type object in manual or fictitious state is done with the predefined semantics given in [Table 4](#). For an object of IEC/REX type in auto state only output objects (BO, DO, AO, BS) are set with #SET LIST command, see [Table 4](#).

Data type:	Integer
Value:	0 Cancelling the selection (IHC) 1 Selection (CBXC)
Default value:	0
Access:	Read, conditional write. It can be written with #SET.
Object types:	BO, AO



SE = 1 may mean that the object is already selected by another operator. The object cannot be selected again until SE is set to 0.

Example:

```
#ERROR CONTINUE
#IF CMD:PSE1 == 0 #THEN #BLOCK
  #SET CMD:PSE1 = 1
  #IF STATUS == 0 #THEN #BLOCK
    #SET CMD:PBO1 = 0
    #IF STATUS <> 0 #THEN !SHOW ERROR "FAILED"
    #BLOCK_END
  #ELSE !SHOW ERROR "SELECTION FAILED"
  #BLOCK_END
#ELSE !SHOW ERROR "ALREADY SELECTED"
```

If not already selected, the command object CMD1 is selected, otherwise an error message "ALREADY SELECTED" is displayed. If the selection succeeded, the command object is set to 0, otherwise an error message "SELECTION FAILED" is displayed. If the set operation did not succeed, an error message "FAILED" is displayed.

5.3.8.4 SP Stop Execution

Writing to this attribute interrupts the RTU200 object command under execution.

Data type:	Integer
Value:	1
Access:	Read, conditional write. It can be written with #SET.
Object types:	BO

Example:

```
#SET CMD:PSP5
```

5.3.9

IEC, REX and DNP specific attributes

The following attributes implement the protocol specific information used by IEC, REX and DNP communication.

These attributes have a predefined semantics only for IEC, REX and DNP stations. For output objects, they are configuration attributes that are sent within ACP messages (see below). For input objects, they are dynamic attributes received in ACP messages. The attributes are included in the snapshot variable lists. To fully support selection and its cancellation in these protocols, the schema presented in [Table 4](#) is to be used.

CT, OG and TY are always sent in the ACP message if corresponding attribute is specified on the #SET list. If they are not on the list, the values in the corresponding database attributes are sent unless they are zero.

Table 4: Schema concerning SCIL interface, process database and ACP messages. SOV stands for Selected OV, a value stored in the database (without a SCIL attribute name)

Action	SCIL	Process database
Select Open	#SET n:PSEi= list(OV='open'[,CT=a][,OG=b][,TY=c])	SE=1, SOV=OV
Select Close	#SET n:PSEi= list(OV='close'[,CT=a][,OG=b][,TY=c])	SE=1, SOV=OV
Select Deactivate	#SET n:PSEi=0 #SET n:PSEi=list(SE=0[,CT=a][,OG=b][,TY=c])	SE=0
Execute Open	#SET n:POV='open' #SET n:POVi= list(OV='open'[,CT=a][,OG=b][,TY=c])	OV='open', SE=0
Execute Close	#SET n:POVi='close' #SET n:POVi= list(OV='close'[,CT=a][,OG=b][,TY=c])	OV='close', SE=0
Execute Deactivate	#SET n:PSEi=0 #SET n:PSEi=list(SE=0[,CT=a][,OG=b][,TY=c])	SE=0

5.3.9.1

OG Originator Identification

Defines the source of the data for both input and output objects. For input objects, this is dynamic information and for output objects a configuration attribute. OG is sent along the ACP message in control messages if its value differs from 0. See also [Table 4](#).

Data type:	Integer
Value:	0 ... 65 535
Access:	No restrictions
Object types:	All

5.3.9.2

QL Command Qualifier

Extra qualifier for commands and various purposes, for both input and output objects. For input objects this is dynamic information, and for output objects a configuration attribute. QL is sent along the ACP message in control messages if its value differs from 0. See also [Table 4](#).

Data type:	Integer
Value:	0 ... 65 535
Access:	No restrictions
Object types:	All

5.3.9.3 TY Type Identification

Defines how the command should be interpreted by the communication unit, (for example, single/double bit command, regulating step command). For input objects this is dynamic information, and for output objects a configuration attribute. TY is sent along the ACP message in control messages if its value differs from 0. See also [Table 4](#). ASDU type or number of received command or indication is written to the TY attribute of the input process object. The ASDU types vary according to the used protocol. For example, ASDU type 45 for IEC60870-5-101 protocol is IEC single command.

Data type:	Integer
Value:	0 ... 65 535
Access:	No restrictions
Object types:	All

5.3.10 OPC Event objects

OPC Event (OE) type process objects are defined to receive event notifications from OPC A&E servers according to the Alarms and Events Custom Interface Standard, Version 1.10, by OPC Foundation.

OPC Event objects may be created only into a unit that is connected to a station object of OAE type. Furthermore, the node (ND) attribute of the station object must denote an OPC_AE type node. On the other hand, an OAE unit may contain only OE type process objects. Because of these restrictions, the following steps must be taken before any OPC Event object may be created in unit u:

1. Create a node object (NODn:B) for the OPC A&E server. The type of the node object is OPC_AE (NODn:BNT = "OPC_AE") and the attribute NODn:BOP describes the OPC A&E server.
2. Create a station object (STAu:B) for unit u. The type of the station is OAE (STAu:BST = "OAE") and the station is linked to node object n (STAu:BND = n).

For details of the node and station objects, see the System Objects manual.

5.3.10.1 Basic functionality

The basic functionality of an OPC Event object is to receive event notifications from an OPC A&E server and store the received data in various attributes of the object.

The name of the event source is defined by the IN (Item Name) attribute of the object. The EH (Event Handling) attribute of the object is left empty if only the basic functionality is used by the application. In this case, the IN attribute must be unique within the unit (UN).

No SYS600 alarm handling is done. The value of the OE attribute is always 0. If history logging is enabled (HE = 1), the received message text (attribute VM) is used as the message text identifier for the SYS600 event list.

Application specific actions may be taken by designing a command procedure run by the associated event channel object (attribute AN, Action Name). All of the OPC Event specific attributes are passed as snapshot variables to the command procedure.

5.3.10.2 Full functionality

Full functionality of OE Event objects includes the basic functionality plus the following:

- Differentiation between the three event types defined by the OPC standard (simple, tracking and condition events)
- Several OPC Event objects for one event source to facilitate different handling of different events from the same source
- Translation of event message texts to numerical values of the process object (attribute OE)
- Alarm handling of condition events
- Full support for SYS600 history database, for example, multilingual application support

The full functionality is obtained via an event handling object linked to the OE object (attribute EH, Event Handling).

The event handling object specifies

1. which type of events the process object is to receive
2. how the numeric value of the process object is deduced from the contents of the event notification
3. how the OPC event is stored in the history database.

There may be up to 200 OE process objects sharing the same IN attribute value (that is, the same OPC event source), provided that each of them is connected to a different event handling object. For example, an OPC event source may send a simple event, a tracking event and two different condition events. Thus, four OE process objects are created to catch the events.

Alarm handling of condition events is enabled simply by defining the alarm class (attribute AC) of the OE process object.

For the details of event handling objects, see [Section 6](#).

5.3.10.3 Undefined event sources

When an event notification with an undefined event source is received, the event channel UNDEF_OPC_EVENT is activated to inform the application of a missing process object (cf. UNDEF_PROC for OA-addressed process objects), see [Section 11](#).

5.3.10.4 Initial state of condition events

There is an inherent problem in the OPC standard related to the initial states of the conditions. This problem should be recognized by anyone doing application engineering of an OPC A&E server.

The standard defines a method called IOPCEventSubscriptionMgt::Refresh, which should be used by every OPC A&E client at start-up to receive the initial states of the conditions. However, this method causes an event notification only for the **active** (and inactive but unacknowledged) conditions, probably to reduce the amount of transmitted data. The reasoning is that the conditions not received in the refresh event notifications are inactive. Therefore, the only thing that the base system can do is to guess that such OE objects are in their normal, non-alarming state (OE = 0) and everything is OK. However, this is not necessarily true: It is possible that there is a **configuration error** in the object. For example, the source

name (attribute IN) may be misspelled or the associated event handling object may specify a condition that the source does not have at all.



There are OPC servers that do not report anything about the event sources currently unreachable by the server. For example, if there are three devices connected to one OPC server and the connection to device 3 is broken for a reason or another, the server may report active conditions from devices 1 and 2 but nothing from device 3. Relying on the standard, the client (here SYS600) draws the conclusion that all the conditions of device 3 are inactive.

To ease the engineering of OE objects, two new SCIL functions have been implemented.

- `OPC_AE_NAMESPACE` browses the name space, that is, event categories, conditions and event sources of any OPC A&E server. The results may be used to construct the corresponding OE process objects and event handling objects in the process database.
- `OPC_AE_VALIDATE` may be used to check the configuration. It cross-checks the process database and the name space of the OPC A&E server and reports any discrepancies found. For details, see the Programming Language SCIL manual.

Another problem related to the initial values of condition events may arise when an OE object is created on-the-fly. The base system tries to read the current state of the condition by using the method `IOPCEventServer::GetConditionState`. However, this method is optional and not all OPC A&E servers have implemented it.



When an OPC Event process object is created, SYS600 may or may not get an initial value for the object depending on the OPC A&E server implementation. If not, the object is updated only when it receives an event, or the states of all condition events are refreshed by SCIL function `OPC_AE_REFRESH`, or the SYS600 OPC A&E client is restarted.

The state of all the condition events may be refreshed by the SCIL function `OPC_AE_REFRESH`, see the Programming Language SCIL manual for details.

5.3.10.5 Setting object status by SCIL

As mentioned above, there are OPC A&E servers that do not report in the refresh or initial notifications the bad status of events whose source device is unreachable. The same may be true when a previously reachable device becomes unreachable, and no event notification is sent for the event sources of the device.

In this case, the SYS600 application may be programmed to set the status of the lost OE process objects to 2 (indicating that the value in the process database may be outdated). Prerequisite to this is that the application is able to deduce the lost device from some event sent by the server. In addition, the application must be able to recognize the OE objects whose event source is located in the lost device.

The SCIL command syntax for setting the OS of a lost OE object is

```
#SET OE_OBJECT:P = LIST(SU = 1, OS = 2)
```

The substitution mechanism is used, the SU attribute value 1 in the process object indicates here that the OS value is manually set. If the object has no value (OS = 10), the command does nothing, the OS stays as 10 and SU as 0.

When the object later receives a genuine event notification from the server, both SU and OS will be automatically reset to 0.

5.3.10.6 Alarm acknowledgement

Active OPC A&E conditions (alarms) are acknowledged by SCIL function `OPC_AE_ACKNOWLEDGE`, see the Programming Language SCIL manual for details.

The function takes four arguments that are passed to the A&E server:

- Acknowledger ID is a text string that identifies the acknowledger
- Comment is a free text supplied by the acknowledger
- Cookie is the value of the CK attribute of the event to be acknowledged
- Active Time is the value of the CQ attribute of the event to be acknowledged

The "Cookie" and "Active Time" arguments are used to identify the event to be acknowledged. These are needed because the semantics of acknowledgement in the OPC specification is more or less tricky: An OPC A&E server may (but is not required to) track alarms, that is, to maintain a queue or list of unacknowledged alarms of a single source, and it may require that all of them be acknowledged. So, if a condition becomes active and then inactive without being acknowledged, and then active again, there may be two alarms to be acknowledged.



The behavior of different OPC A&E servers may vary. Some servers do track alarms, others do not. The ones that do may put a limit for the length of the unacknowledged alarm list. The behavior in case the limit is exceeded is totally vendor-specific. Consequently, the documentation of the used A&E server must be carefully studied when an application that acknowledges A&E alarms is designed.

By default, the SYS600 alarm handling and the OPC A&E alarm handling are totally independent. SYS600 alarms are acknowledged by setting the AR attribute of the process object to 1. OPC A&E alarms are acknowledged by SCIL function `OPC_AE_ACKNOWLEDGE`.

If the application wants to couple these two types of alarms, the AA (Auto Acknowledge) field of the OP attribute of the A&E server's node object is set to 1. In this case, setting the AR attribute of the process object generates an automatic acknowledgement to the A&E server, provided that the process object is related to a condition event and that condition is currently active.

The four acknowledgement arguments discussed above have the following values:

- The name of the MicroSCADA user who acknowledges the alarm is used as the Acknowledger ID.
- Comment is a null string.
- Cookie and Active Time are the ones received in the latest event notification.

SYS600 does not maintain any list of unacknowledged A&E alarms. Consequently, it is possible that 'old' alarms cannot be acknowledged using the Auto Acknowledge feature.

5.3.10.7 OPC Event attributes

The OPC Event specific attributes are described below in alphabetical order.

All of them are dynamic, read-only attributes that have been assigned a value from the event notification sent by the connected OPC A&E server. The corresponding notification member names are given in attribute descriptions as defined by the Alarms and Events Custom Interface Standard.

Besides the attributes common to all process object types and the OE specific ones described below, operation counting attributes CE, CL, CO and CV are supported by OPC Event objects, see [Section 5.2.15](#) and [Section 5.3.5](#).

AK OPC Ack Required

This attribute indicates whether the related condition requires acknowledgement of the event.

Data type:	Integer	
Value:	0	Acknowledgement not required
	1	Acknowledgement required
Access:	Read-only	
Member:	bAckRequired	

The SCIL function OPC_AE_ACKNOWLEDGE may be used to acknowledge the event in the A&E server, see the Programming Language SCIL manual.

CK OPC Cookie

Server defined cookie associated with the event notification.

Data type:	Integer	
Value:	OPC server specific values	
Access:	Read-only	
Member:	dwCookie	

This value along with the CQ attribute value is used to identify the event to be acknowledged with the SCIL function OPC_AE_ACKNOWLEDGE.

CM OPC Change Mask

A bit mask indicating which properties of the condition have changed. See the OPC Alarms and Events standard and the OPC server documentation for the possible values of this attribute.

Data type:	Integer	
Value:	OPC server specific values	
Access:	Read-only	
Member:	wChangeMask	

CN OPC Condition Name

The name of the condition related to the event notification.

Data type:	Text	
Value:	OPC server specific	
Access:	Read-only	
Member:	szConditionName	

CQ OPC Cookie Time

Opaque 8-byte data to identify an event.

Data type:	Byte string	
Value:	The value of the OQ attribute in a format required by the OPC_AE_ACKNOWLEDGE function	
Access:	Read-only	
Member:	ftActiveTime	

This value, along with the CK attribute value, is used to identify the event to be acknowledged with the SCIL function OPC_AE_ACKNOWLEDGE.

ID OPC Actor ID

The actor ID of a tracking event or the acknowledger ID of a condition event.

Data type:	Text
Value:	OPC server specific
Access:	Read-only
Member:	szActorID

The value of this attribute is used as the user name (event attribute US) in the post-processing of the event.

NS OPC New State

A bit mask specifying the new state of the condition.

Data type:	Integer
Value:	0 ... 7, any combination of the following bit values
	1 OPC_CONDITION_ENABLED
	2 OPC_CONDITION_ACTIVE
	4 OPC_CONDITION_ACKED
Access:	Read-only
Member:	wNewState



There are OPC A&E servers that define vendor-specific bits in the mask, violating the OPC standard.

OE OPC Event

This attribute contains the numeric value of the OE type process object. This is the main attribute of the object (OV, Object Value).

Data type:	Integer
Value:	0 ... 255
Access:	Read-only

The value of this attribute is deduced from the contents of the OPC event notification. If no event handling object is associated with the object (EH = ""), the value is always 0. Otherwise, the following rules are applied:

- For simple and tracking events, the event message list (attribute EM of the associated event handling object) is searched for the message received in the event notification (attribute VM). If the event message matches the first message of the list, OE is set 0. If it matches the second message, OE is set to 1, etc. If no match is found, OE is set to the length of the list, that is, the last 'valid' value + 1. If the EM attribute of the event handling object is not given, OE is always set to 0.
- For condition events, OE is set to 0 if the condition is inactive.
- If the condition is active and the condition is a simple condition (having no sub-conditions), OE is set to 1.
- If the source is in a sub-condition state, the OE is set to the ordinal number of the current sub-condition. The sub-conditions are numbered by the order they are listed in the SN (Subcondition Names) attribute of the associated event handling object.

OQ OPC Active Time

Time that the condition became active or the time of the transition into the current sub-condition.

Data type:	List
Value:	Qualified time, attributes:
	CL Time
	MS Integer, milliseconds
	DS Boolean, daylight saving (always FALSE)
Access:	Read-only
Member:	ftActiveTime

QU OPC Quality

Quality associated with the condition state. This 16-bit value consists of 3 separate bit fields. See the OPC Alarms and Events standard and the OPC server documentation for the possible values of this attribute.

Data type:	Integer
Value:	0 ... 65 535, OPC server specific values
Access:	Read-only
Member:	wQuality

If the value of this attribute is not one of the 'good' values, the OS (Object Status) attribute of the process object is set to 2.

SE OPC Severity

Event severity.

Data type:	Integer
Value:	1 ... 1000, OPC server specific values
Access:	Read-only
Member:	dwSeverity

SN OPC Subcondition Name

The name of the current sub-condition.

Data type:	Text
Value:	OPC server specific
Access:	Read-only
Member:	szSubconditionName

VC OPC Event Category

The category ID of the event.

Data type:	Integer
Value:	OPC server specific code
Access:	Read-only
Member:	dwEventCategory

VM OPC Event Message

Event notification message describing the event.

Data type:	Text
Value:	OPC server specific
Access:	Read-only
Member:	szMessage

VQ OPC Event Time

Time of the event occurrence in UTC time.

Data type:	List
Value:	Qualified time, attributes:
	CL Time
	MS Integer, milliseconds
	DS Boolean, daylight saving (always FALSE)
Access:	Read-only
Member:	ftTime

VT OPC Event Type

The type of the event.

Data type:	Integer
Value:	The OPC event type
	1 OPC_SIMPLE_EVENT
	2 OPC_TRACKING_EVENT
	4 OPC_CONDITION_EVENT
Access:	Read-only
Member:	dwEventType

5.3.11 File transfer attributes

The term File Transfer is used to mean any bulk data transfer between SYS600 and a station. A file is a sequence of bytes of any length with no interpretation of its contents. The implementation of file transfer in SYS600 base system is protocol independent, the interpretation of the contents of a file is done with SCIL.

In SYS600, a file is represented by a disk file and identified by its disk file name. In a station, a file is typically, but not necessarily, a memory segment in RAM and the identification or naming is protocol dependent. The ID is handled in SYS600 as a byte string with no interpretation.

File transfer is always initiated by SYS600, spontaneous file transfer from station to SYS600 is not supported. File transfer is done asynchronously. It is initiated by SCIL, but the SCIL command does not wait for the completion of the transfer.

The FT (File Transfer) process object type implements the file transfer functionality. The following functions are supported:

- Receiving (uploading) a file from a station.
- Sending (downloading) a file to a station.
- Browsing the file hierarchy of a station.
- Reading file attributes from a station.
- Deleting a file or directory in a station.

The process objects of this type have all the same common attributes as "real" process objects. Especially, the post-processing attributes, such as EE and AE, may be used to report the completion of the transfer to the application. The completion of the transfer is indicated by a change of the value of FT attribute. The address attributes OA and OB have no meaning in conjunction with FT objects, OA should be set to 0, if set at all. There may be any number of FT objects connected to one station, for example each configured to a specific download/upload.

The attributes specific to File Transfer process objects are described below.

5.3.11.1 DC Directory Contents

Uploaded contents of a directory of a station.

Data type:	List of vector attributes												
Value:	Attributes and their element types: <table><tr><td>TYPE</td><td>Text value "FILE" or "DIRECTORY"</td></tr><tr><td>ID</td><td>Byte string, the file's ID in the station</td></tr><tr><td>NAME</td><td>Text, the file's name in the station</td></tr><tr><td>CREATION_TIME</td><td>Time value</td></tr><tr><td>LENGTH</td><td>Integer, length of the file as bytes</td></tr><tr><td>AUXILIARY</td><td>Byte string value, station specific auxiliary information</td></tr></table>	TYPE	Text value "FILE" or "DIRECTORY"	ID	Byte string, the file's ID in the station	NAME	Text, the file's name in the station	CREATION_TIME	Time value	LENGTH	Integer, length of the file as bytes	AUXILIARY	Byte string value, station specific auxiliary information
TYPE	Text value "FILE" or "DIRECTORY"												
ID	Byte string, the file's ID in the station												
NAME	Text, the file's name in the station												
CREATION_TIME	Time value												
LENGTH	Integer, length of the file as bytes												
AUXILIARY	Byte string value, station specific auxiliary information												
	Attributes NAME, CREATION_TIME, LENGTH and AUXILIARY are returned only if supported by the station.												
Access:	Read-only												

Example:

```
UPLOAD = FT:PDC10
NAME_1 = UPLOAD.NAME(1)
TYPE_1 = UPLOAD.TYPE(1)
```

5.3.11.2 FF File Function

The function to be performed on a file.

Data type:	Integer										
Value:	0 ... 4: <table><tr><td>0</td><td>None</td></tr><tr><td>1</td><td>Send a file</td></tr><tr><td>2</td><td>Receive a file</td></tr><tr><td>3</td><td>Read directory</td></tr><tr><td>4</td><td>Delete a file or directory</td></tr></table>	0	None	1	Send a file	2	Receive a file	3	Read directory	4	Delete a file or directory
0	None										
1	Send a file										
2	Receive a file										
3	Read directory										
4	Delete a file or directory										
	Setting FF to 0 cancels the ongoing transfer										
Default value:	0										
Access:	Read, conditional write. May be written only when switch state is AUTO										

5.3.11.3 FN File Name

The tag of the disk file to be sent or received.

Data type:	Byte string
Value:	File tag
Access:	No restrictions

Example:

```
#SET FT:PFN10 = FM_FILE("C:\sc\conf.txt")
```

5.3.11.4 FP File Transfer Progress

Displays the number of bytes transferred between NET and the station during current transfer (FT = 1) or last transfer (FT = 4). Valid only when FF = 1 or FF = 2, otherwise 0.

Data type:	Integer
Unit:	Bytes
Access:	Read-only

5.3.11.5 FT File Transfer

Displays the state of transfer.

Data type:	Integer
Value:	0 ... 4:
	0 Idle
	1 Transfer in progress
	2 Cancelled by the user
	3 Aborted by station or communication error
	4 Ready
Default value:	0
Access:	Read-only

5.3.11.6 ID Identification

The file's ID in the station. An ID may be a fixed value known by the operator, or it may be retrieved by reading the directory structure of the station. A zero length byte string denotes the root of the directory structure. The value is station type (or protocol) specific. When creating an ID with SCIL, differences in CPU architecture (especially byte-ordering) may have to be taken care of.

Data type:	Byte string
Value:	Station type specific
Access:	No restrictions

Example:

```
#SET FT:PID10 = PACK_STR(VECTOR(97,234), "byte_string", 1, "big_endian")
```

5.3.11.7 ST Status

A SCIL status code giving more information when the transfer is aborted (FT = 3).

Data type:	Integer
Value:	A SCIL status code
Default value:	10
Access:	Read-only

5.3.11.8 Examples of using file transfer

The following sequence may be used in a SCIL program to receive a file from a station or to send a file to a station:

1. Set attributes FN and ID (in any order) to identify the file both in SYS600 and in the station. Setting one of these attributes automatically sets FT to 0 (idle state).
2. Set FF to 1 or 2 to initiate the transfer. This step generates a SCIL error if the station cannot accept the request (invalid ID, not enough memory to receive the file, etc.).
3. Wait for the completion of the transfer or cancel it by setting FF to 0.

The following sequence may be used to browse the directory structure in a station:

1. Set ID to zero length byte string (indicating the root directory of the station).
2. Set FF to 3 to initiate the transfer.
3. Wait for completion (or cancel by setting FF to 0).
4. Read the result from the DC attribute.
5. Pick up the ID of desired file and start a file transfer, or pick up the ID of a subdirectory, set it to the ID attribute and repeat steps 2 to 5.

The following sequence may be used to read attributes of a file:

1. Set attribute ID to identify the file in a station.
2. Set FF to 3 to initiate the query.
3. Wait for the completion.
4. Read the result from the DC attribute (DC contains the attributes of the file as one-element long vectors).

The following sequence may be used to delete a file in a station:

1. Set ID attribute to identify the file in the station.
2. Set FF to 4.
3. Wait for the completion.

5.3.12 Network topology attributes

5.3.12.1 LE Level Enumeration

The calculated level of a network topology object.

Data type:	Integer
Value:	0 .. n, the enumeration of the level. The maximum value n depends on the application specific configuration of the topology schema. In the POWER schema, up to 16 levels (0 .. 15) may be specified.
Access:	Read-only
Object types:	NT

5.3.12.2 LP Loop State

The calculated loop state of a network topology object.

The possible loop states depend on the topology schema. The values listed below correspond to the POWER schema.

Data type:	Integer	
Value:	0	No loop
	1	Topological loop
	2	Power loop
	3	Topological and power loop
Default value:	0	
Access:	Read-only	
Object types:	NT	

5.3.12.3 LV Level Value

The calculated nominal level of a network topology object.

The levels and their nominal values depend on the topology schema.

Data type:	Real
Value:	In the POWER schema, the nominal voltage level as kilovolts.
Default value:	0.0
Access:	Read-only
Object types:	NT

5.3.12.4 LX Level Text

The translated textual name of the calculated level of a network topology object.

The levels and their names depend on the topology schema.

Data type:	Text
Value:	The localized name of the level
Access:	Read-only
Object types:	NT

5.3.12.5 ND Network Topology Data

Auxiliary calculated topology data for visualization clients.

The structure and contents of this attribute are defined by the topology schema. In the POWER schema, the attribute has been defined as follows:

Data type:	Vector	
Value:	Each element describes one feed type	
Element type:	Vector of 3 integer elements:	
	1	The network object subtype (NS) of the feeder object
	2	The status of the feed (0 = certain, 1 = uncertain)
	3	Color index for coloring by source

Table continues on next page

Default value:	Empty vector
Access:	Read-only
Object types:	NT

5.3.12.6 NF Network Feeds

The network topology (NT) objects that feed this object, including earths.

The structure and contents of this attribute are defined by the topology schema. In the POWER schema, the attribute has been defined as follows:

Data type:	Vector
Value:	Each element describes one feeder
Element type:	List of 3 attributes: LN Logical name IX Logical index STATUS 0 = certain, 1 = uncertain When STATUS = 1, the topology calculation algorithm cannot tell for sure whether this source feeds the object or not (due to missing or faulty process data).
Default value:	Empty vector
Access:	Read-only
Object types:	NT

When an NT object is multiply earthed, only one of the earths is listed.

5.3.13 Event history attributes

Each event in the history database contains a snapshot of all attributes of the process object. Additionally, some information related to the event itself is stored in the database as pseudo attributes described below. They cannot be accessed using the standard process object notation, but they can be used just like ordinary process object attributes in arguments of HISTORY_DATABASE_MANAGER function.

Attributes CA (Changed Attribute), MX (Message Text) and US (User Name) are available through the OPC Data Access Server as well, see the OPC Server manual for details.

For more information about commands and functions used for queries, see the Programming Language SCIL manual and for information on how to configure the event history see the System Configuration manual. Process object attributes related to history logging are described in [Section 5.2.10](#).

5.3.13.1 CA Changed Attribute

The name of the process object attribute that changed and caused the event logging.

Data type:	Text
Value:	Two character attribute name
Access:	Read-only

5.3.13.2 ED Event Daylight Saving

Indicates whether daylight saving time was in use at the time of event (ET).

Data type:	Integer	
Value:	0	Not known
	1	Daylight saving time not in use
	2	Daylight saving time in use
Access:	Read-only	

5.3.13.3 EM Event Time Milliseconds

The milliseconds of the time of the event. For true process events, EM is equal to RM.

Data type:	Integer	
Value:	0 ... 999	
Unit:	Milliseconds	
Access:	Read-only	

5.3.13.4 EQ Qualified Event Time

The time of the event as qualified time. For true process events, EQ is equal to RQ. This attribute combines the information of ET, EM and ED.

Data type:	List	
Value:	Qualified time, attributes	
	CL	Time
	MS	Integer, milliseconds
	DS	Boolean, daylight saving
Access:	Read-only	

5.3.13.5 ET Event Time

The time of the event, in a resolution of one second. For true process events, ET is equal to RT.

Data type:	Time
Access:	Read-only

5.3.13.6 EX Event Comment Text

An event comment text given by the operator.

Data type:	Text	
Value:	Any text, up to 255 characters	
Default value:	""	
Access:	Can be read and written by HISTORY_DATABASE_MANAGER	

5.3.13.7 HD History Logging Daylight Saving

Indicates whether daylight saving time was in use when the event was written into the history database (HT).

Data type:	Integer	
Value:	0	Not known
	1	Daylight saving time not in use
	2	Daylight saving time in use

Access: Read-only

5.3.13.8 HM History Logging Time Milliseconds

The milliseconds of the time when the event was written into the history database.

Data type:	Integer	
Value:	0 ... 999	
Unit:	Milliseconds	
Access:	Read-only	

5.3.13.9 HQ Qualified History Logging Time

The time when the event was written into the history database (as qualified time). This attribute combines the information of HT, HM and HD.

Data type:	List	
Value:	Qualified time, attributes	
	CL	Time
	MS	Integer, milliseconds
	DS	Boolean, daylight saving
Access:	Read-only	

5.3.13.10 HT History Logging Time

The time when the event was written into the history database.

Data type:	Time
Access:	Read-only

5.3.13.11 MX Message Text

A descriptive text of the event that caused the logging.

The text is obtained from the event handling object referenced by the EH attribute of the process object.

Data type:	Text	
Value:	A descriptive text of the event in the current language	
Access:	Read-only	

5.3.13.12 PV Previous Value

The value of the process object before the event.

Data type:	The data type of the OV attribute of the object
Value:	The value of the OV attribute before the event
Access:	Read-only

5.3.13.13 US User Name

The name of the MicroSCADA user who caused the logging.

Data type:	Text
Value:	MicroSCADA user name
Access:	Read-only

5.4 Defining process objects

When a new process object of predefined type is defined, some definition attributes are obligatory, others are optional and some attributes are not valid depending on the object type, see [Table 2](#). The following attributes are the minimum requirements:

LN	Logical name
IX	Index (not for user-defined object types)
PT	or any of the attributes AI, AO, BI, BO, BS, DI, DO, DB, PC, FT, OE.
In addition to the three attributes mentioned above, real objects with process communication require the address attributes:	
UN	Unit
OA	Object Address
OB	Object Bit Address (some process object types only)
Process objects located in an external OPC server are addressed by	
IN	Item Name
All analogue objects require, if IR = 0:	
SN	Scale Name

For example, the following optional features can be defined as follows:

- Alarm generating objects: the alarm handling attribute AC (Alarm Class), alarm limits or AG, LA, the SZ attribute for SCADA supervision.
- Monitor alarm: the PI and PD attributes.
- Automatic printing: the PF and LD attributes.
- Event channel: at least the AN and AE attributes.
- Event object (event controlled updating in pictures): the EE attribute.
- History logging (including the object in the event list): the HE attribute.
- Switch state AUTO: the SS attribute.
- Taking the object into use: the IU attribute.

5.4.1 Examples

Example 1:

```
#CREATE PROC:P1 = LIST(AI = 0, SN = "S")
```

An analog input object is created with the name PROC, index 1, analog input value 0 and scale name "S".

Example 2:

```
#LOCAL P = FETCH(0, "P", "OLD", 1)
P.OA = P.OA + 1
#CREATE NEW:P1 = P
```

The new process object is otherwise identical to the old one, but its object address is changed.

5.5 Process object group attributes

This section describes the attributes of process object groups.

5.5.1 CC Control Supervision Configuration

Configuration data for control supervision functionality.

Control supervision functionality generates an alarm when control of a device fails, that is, indication of the expected new state is not received within a specified period of time. During that period, alarms and events of the indication object are blocked.

Data type:	List	
Value:	The following attributes:	
	AC	Alarm Class Integer 0 ... 7. When AC is 0, the control supervision functionality is not generating an alarm.
	RC	Receipt Requirement Optional, integer 0 or 1. When RC is 1 (default value), the generated alarm must be acknowledged.
	AD	Alarm Delay Milliseconds to wait for expected state
	MT	Message Text Id Text id of localizable event text for failed control
	II	Indication Index Index of the input process object indicating the state of the controlled device. The type of the PO must be BI, DB, DI or AI. The switch state of the object may be MAN, AUTO or FICTITIOUS.
	XI	Execute Index Index of the output PO to execute the control. PO type BO, DO or AO. The switch state of the object may be MAN, AUTO or FICTITIOUS.
	XO	Execute Open Index Index of the output PO to execute OPEN control
	XC	Execute Close Index Index of the output PO to execute CLOSE control
	SO	Select Open Index Index of the output PO to select OPEN control
	SC	Select Close Index Index of the output PO to select CLOSE control
	VO	Value to Open Integer, the value assigned to XI to execute or select OPEN control
	VC	Value to Close Integer, the value assigned to XI to execute or select CLOSE control
	VX	Value to Execute Integer, the value assigned to XI to execute the control
Access:	No restrictions	

The subattributes AC, AD, MT and II are mandatory. The allowed combinations of the rest of subattributes depend on the way how the control commands for the device are implemented in the IED. The supported subattribute combinations are

- II, XI
- II, XO, XC
- II, XI, SO, SC
- II, XI, VO, VC
- II, XI, VO, VC, VX

Example 1:

Suppose that process object DOOR:P1 is a binary input object that receives the position of a door (0 = open, 1 = closed) and DOOR:P2 is a binary output object for the door control.

The most simple control supervision configuration could be made by the following command:

```
#SET DOOR:PCC = LIST(AC=1, AD=10000, MT="Door opening/closing failed",
II=1, XI=2)
```

When the door is closed by setting DOOR:PBO2 to 1, a class 1 alarm is generated if the door won't close within 10 seconds (DOOR:PBI1 won't change to 1).

5.5.2 CD Configuration Data

Configuration data of the process group.

Data type:	List
Value:	Any list
Access:	No restrictions

The list may contain any number of attributes of any type. The total size of the contained data is limited to approximately 64 kB.



This attribute is reserved for system (LIBxxx) use. It should not be accessed by the application software.

5.5.3 GA Group Alarm

Indicates whether any of the process objects of the group is alarming.

Data type:	Integer
Value:	0 None of the process objects of the group is alarming
	1 At least one process object of the group is alarming
Access:	Read-only

5.5.4 GB Group Blockings

Summary of blockings of the process objects in the group.

Data type:	List
Value:	The following integer vector attributes:
	AB Indexes of process objects whose alarms are blocked
	CB Indexes of output process objects whose control is blocked
	HB Indexes of process objects whose history (event list) is blocked
	PB Indexes of process objects whose event printouts are blocked
	UB Indexes of input process objects whose updating from the process is blocked
	XB Indexes of process objects whose event channel activations are blocked
Access:	Read-only

5.5.5 GC Group Comment

A freely chosen text.

Data type:	Text
Value:	Any Unicode text, up to 255 characters
Access:	No restrictions

5.5.6 GS Group Alarm State

Summary of alarms and warnings of the process objects in the group.

Data type:	List
Value:	The following integer attributes:
	ACTIVE_NOACK Number of active alarms in the group that do not require an acknowledgement (AL = 1, AR = 1, RC = 0)
	ACTIVE_ACKED Number of acknowledged active alarms in the group (AL = 1, AR = 1, RC = 1)
	ACTIVE_UNACKED Number of unacknowledged active alarms in the group (AL = 1, AR = 0, RC = 1)
	FLEETING Number of fleeting alarms in the group (AL = 0, AR = 0, RC = 1)
	LOW_WARNINGS Number of analog input (AI) objects in the group that are in LOW WARNING state (AC > 0 and AZ = 3)
	HIGH_WARNINGS Number of analog input (AI) objects in the group that are in HIGH WARNING state (AC > 0 and AZ = 4)
Access:	Read-only

If an analog input object is in a warning state and has a fleeting alarm, it is regarded as FLEETING until the alarm has been acknowledged.

5.5.7 GT Group Type

An arbitrary type number defined by the application.

Data type:	Integer
Value:	0 ... 255
Default value:	0
Access:	No restrictions

5.5.8 LF Logical Format

The name of the logical format picture, that is, the format that prints the group with command #LIST, see [Section 4](#).

The name of the group is passed as an argument variable LN to the format picture. This way, several different objects can use the same logical format picture.

Data type:	Text
Value:	Picture name, up to 10 characters
Access:	No restrictions

5.5.9 LN Logical Name

The logical name of the process object group.

Data type:	Text
Value:	Object name, up to 63 characters
Access:	Read-only, configurable

5.5.10 PV Process Views

Names of process views displaying the process group.

Data type:	Text vector
Value:	Up to 10 process view names, up to 255 characters each
Default value:	Empty vector
Access:	No restrictions



The base system does not maintain the value of this attribute. It is the responsibility of the application to keep the value up-to-date.

5.5.11 ZT Modification Time

The time when the object group was created or modified.

Data type:	Time
Access:	Read-only

This attribute is set by the main program when the object group is created and each time the object group is updated by the #MODIFY command (for example, by the process object group definition tool).

Section 6 Event handling objects

6.1 About this section

This section describes the event handling objects and their attributes:

Section 6.2	General : Different types of event handling objects, their basic properties, use and function, etc.
Section 6.3	Common event handling object attributes : Attributes listed and described in alphabetic order.
Section 6.4	OPC Alarms & Events server attributes : Attributes listed and described in alphabetic order.
Section 6.5	OPC Alarms & Events client attributes : Attributes listed and described in alphabetic order.
Section 6.6	Defining event handling objects using SCIL : An example.
Section 6.7	Predefined event handling objects

6.2 General

6.2.1 Different types of event handling objects

There are currently two types of event handling objects implemented:

1. The SYS type event handling objects define the event handling of conventional process objects of SYS600.
2. The AEC (Alarms and Events Client) type event handling objects define the event handling of OPC Event (OE) type process objects, which receive their data from an OPC Alarms and Events (A&E) Server via an OPC Alarms and Events Client of SYS600.

The AEC type objects are extensions to SYS type objects. They have the same functionality as the SYS type objects and additional functionality related to the OPC standard.

6.2.2 Use

Event handling objects define texts associated to the states of process objects and transitions from one state to another (events).

The texts are not defined directly in the event handling objects, only the references to the texts. These references, called text identifiers, are used at run-time to translate the texts into the current language. See the Programming Language SCIL manual, the Language Functions section, for the details of the translation process.

In theory, all the information contained in the event handling objects could be stored in the process objects. However, by grouping functionally similar process objects to classes and defining one event handling object for the class saves a lot of engineering work. The reference to the event handling object is given in the EH (Event Handling) attribute of a process object.

If the EH attribute of a process object is undefined (empty), a default event handling object is applied. Each process object type and subtype has its own default event handling object. There are two scopes of defaults:

- Application default event handling objects are application defined defaults that are first applied. See [Section 6.8](#) for details.
- Predefined event handling objects are system wide hardcoded defaults, which are applied when application default object has not been defined. see [Section 6.7](#) for details.

In future releases of SYS600, some other properties related to the event handling may be implemented by new attributes of event handling objects. Therefore, it is recommended that distinct event handling objects are defined for logically unrelated process object classes, even if the state and event texts happen to match by chance.



No event handling objects used in conjunction with user defined (free type) process objects are implemented.

6.2.3 Common functionality

The event handling objects define the text identifiers for descriptions of various object states and events (meaning roughly the transitions between states).

The object state often refers to the object value (OV) attribute of a process object. However, a process object has many other states as well: It may be blocked (BL), its value may be substituted (SB) or the sensor may be FAULTY (OS = 1), etc. There is a set of predefined event handling objects to define the handling of these additional object states, see [Section 6.7](#). The event handling objects created by the application always refer to the OV related object state.

The object state is an integer that is used as an index to the state text array (attribute ST) and to the event message text matrix (attribute MT). The state is calculated in three steps as follows:

1. The **object value**, which can be either an integer or a real number, is first defined according to the process object type:
For the analog input (AI) objects that have an alarm class, the value of the AZ attribute defines the object value.
For all other types, the OV attribute (BI, BO, DB etc.) is directly used as the object value.
2. The value formula (attribute VF) is applied to the object value resulting in an **event value**, which is an integer. The formula defines simple arithmetic operations to be performed on the object value and optionally on another, user-defined attribute.
If no formula is defined, the object value is the event value.
3. The **object state** is calculated by comparing the event value (rounded to the nearest integer) to the attributes VC (Value Count) and VL (Value Low), which define the set of all object states.
If the event value falls outside the defined object states, the object state is undefined. The resulting state and message texts will be empty.
For efficient handling of objects that may have a lot of different values, a range of event values may be defined to represent one object state (see the VC attribute below).

The event value calculated in step 2 may be used to format the translated state and the message texts.

In the state texts, which are used to give the textual description of the value of an process object (the attribute SX of the process object), the numeric event value may be referred to in the translation by using the following notation:

Current \V2\ A

The event value, expressed with 2 decimals, is inserted in the translation. The resulting text might then be:

Current 12.34 A

If the number of decimals is not given, the value is displayed with roughly six significant digits.

In the event message texts (and state texts that are used as message texts), also the previous event value is available. For example, the translation

Setpoint changed from \P1\ to \V1

might generate the text

Setpoint changed from 5.1 to 5.5

An alternative way to format the texts is to use SCIL Data Derivation Language (SDDL). When a translated text (State Text SX or Event Message Text MX) begins with "SDDL:" (case insensitive), the rest of text is taken as an SDDL expression that is evaluated in the context of the process object to format the text. Any attribute of the process object may be used to format the text. However, recursive usage is forbidden: State text may not use SX attribute and event message text may not use MX attribute. In case of event message texts, the pseudo-attribute PV (for Previous Value) may be used to refer to the object's OV value before the event. For details of SDDL, see the Programming Language SCIL manual.

As an example, consider the High Alarm event of an analog input object. The default English translation of the event message text is simply "High alarm". To add some additional information to the message, the following translation could be defined:

SDDL: "High alarm, value " + dec(AI, 0, DP) + ", alarm limit " + dec(HI, 0, DP) + " " + ST

The expression takes advantage of attributes AI (Analog Input), HI (High Input), DP (Decimal Places) and ST (Engineering Unit) of the object and uses SCIL function dec to format the text. The result might be:

High alarm, value 24.12, alarm limit 24.00 kV

There is a set of predefined event handling objects for object value related states and events. One of these objects is applied when the EH attribute of a process object is not defined. For details, see [Section 6.7](#).

6.2.4 AEC functionality

The AEC event handling objects specify:

1. which events the process objects connected to the event handling are to receive
2. how the numeric values of process objects are deduced from the contents of the event notifications
3. how the OPC events are shown in the SYS600 event list

Essentially, the 3rd item includes the functionality of SYS type event handling objects.

6.2.5 Storage

The maximum number of event handling objects in an application is 65 535. The objects are stored in the process database file APL_PROCES.PRD. When the application is WARM or HOT, the objects are stored in the global memory pool of SYS600 as well (see the System Objects manual).

6.2.6 Event handling object notation

Event handling objects and their attributes are accessed from SCIL with the following object notation (see also [Section 4](#)):

name:{application}H{attribute}{(index)}

or

name:{application}H{attribute}{index}

where

'name'	is the name of the object
'application'	is the logical application number
'attribute'	is the attribute name
'index'	is an index or index range

6.3 Common event handling object attributes

6.3.1 CX Comment Text

A freely chosen text.

Data type:	Text
Value:	Any Unicode text, up to 255 characters
Access:	No restrictions

6.3.2 HT Event Handling Type

Specifies the type of the event handling object.

Data type:	Text keyword
Value:	One of following:
	"SYS" Basic event handling object
	"AEC" OPC A&E Client event handling object
Default value:	"SYS"
Access:	Read-only, configurable

6.3.3 LN Logical Name

The name of the event handling object.

Data type:	Text
Value:	Object name
Access:	Read-only, configurable

6.3.4 MT Message Texts

The text identifiers of the event message texts.

Data type:	Text vector
Length:	0 or 2, if VC = 0. 0 or n * n, if VC > 0. "n" is the number of object states. The maximum length is 65 536.
Indexing:	By new and previous object state, see below
Element value:	Text identifier
Default value:	Empty vector
Access:	Read-only, configurable

If the value of the VC (Value Count) attribute is 0, the MT attribute typically defines two text identifiers, the first for the "Value changed message and the second for the "Value updated but not changed" message. The attribute may be empty, provided that the object has a non-empty ST attribute (of length 1). In this case, the ST attribute is used to generate the event message texts.

If the value of the VC attribute is non-zero, the MT attribute may be either empty or a vector of n * n text identifiers, where "n" is the number of distinct object states (see attribute VC). The maximum length of the MT attribute is 65 536 texts. This limit implies that the MT attribute may be used only when the state count n is less than or equal to 256.

When the MT attribute is empty, the state texts (attribute ST) are used as event message texts as well, that is, the description of the new object state is used as the description of the event. The state texts are used as message texts also in cases where the previous state of the process object is not known, for example when the value of the object is updated for the first time after the application start-up.

When the MT attribute is defined, the event message text depends on both the new and the previous state of the object. The MT attribute contains an n x n matrix of text identifiers.

The following table illustrates the indexing and semantics of the MT attribute. It is assumed that VC = 2 and VL = 0. The text identifiers are named as "EVENT_p_TO_n", where p is the previous state and n is the new state.

	New state = 0	New state = 1
Old state = 0	(1) "EVENT_0_TO_0"	(2) "EVENT_0_TO_1"
Old state = 1	(3) "EVENT_1_TO_0"	(4) "EVENT_1_TO_1"

6.3.5 MX Translated Message Texts

The event message texts translated into the current language.

Data type:	Text vector
Length:	0 or 2, if VC = 0. 0 or n * n, if VC > 0. "n" is the number of object states. The maximum length is 65 536.
Indexing:	By new and previous object state, see below
Element value:	Event message text in the current language, or an SDDL expression which evaluates to the event message text
Access:	Read-only, not returned by the FETCH function

For the indexing and the semantics of the attribute, see the MT attribute.

6.3.6 ST State Texts

The text identifiers of the object state texts.

Data type:	Text vector
Length:	The number of distinct object states, 0 ... 65 536
Indexing:	Object state, see below
Element value:	Text identifier
Default value:	Empty vector
Access:	Read-only, configurable

The first element of the vector defines the text identifier corresponding to the lowest value of the object state, the last element corresponds to the highest value. For object state numbering, see the VC attribute.

Even if no distinct object states have been defined (VC = 0), the ST attribute may be used to define a text that is related to the value of the object. In this case, one text identifier is defined (the length of the ST vector is 1), and the event value is inserted in the translation, see [Section 6.2](#).

6.3.7 SX Translated State Texts

The object state texts translated into the current language.

Data type:	Text vector
Length:	The number of distinct object states, 0 ... 65 536
Indexing:	Object state, see attribute ST
Element value:	Object state text in the current language, or an SDDL expression which evaluates to the state text
Default value:	Empty vector
Access:	Read-only, not returned by the FETCH function

6.3.8 VC Value Count

The number or a list of the distinct event values that the process object may have.

Data type:	Integer or a vector
Value:	Integer 0 ... 65 536 or a vector of up to 65 536 distinct values or ranges, see below
Default value:	0
Access:	Read-only, configurable

Typically, the VC attribute is defined as an integer. The possible object state values are then VL, VL + 1, ..., VL + VC -1.

For AEC event handling objects, the value of this attribute follows the length of the EM (Event Messages) attribute (SIMPLE and TRACKING events), or the length of the SN (Subcondition Names) attribute (CONDITION events), and is not directly configurable.

The vector form, available only in SYS event handling objects, is used when the space of event values is sparse or there are a lot of distinct values. For example, an object may take only the values 0, 100, 101, 102 and 199. It would be extremely clumsy to define 200 state texts and 40 000 message texts for the events.

The vector lists all the distinct values and value ranges that the event value may have. The distinct values are given as integers, the ranges as two-element integer vectors. The values within a range are considered to represent the same object state. Consequently, they share the corresponding state text and message texts. The values must be listed in ascending numerical order.

When the vector form is used, the VL (Value Low) attribute has no meaning, because the first element of the vector already defines the lowest possible value.

For the definition of the event value, see [Section 6.2](#).

Example:

The VC attribute of the example case above could be given as:

VECTOR (0, (100, 102), 199)

The corresponding 3-state ST attribute might be:

VECTOR ("TEXT_Low", "TEXT_Intermediate", "TEXT_High")

6.3.9 VF Value Formula

The formula for scaling the object value or calculating the event value out of the object value and an additional attribute.

Data type:	Text
Value:	Up to 255 characters, see below
Default value:	""
Access:	Read-only, configurable

The full syntax of the formula is:

$+/- m * OV / n +/- p * aa / q$

where

m, n, p and q	are integer constants
OV	refers to the object value (either the OV or the AZ attribute of the process object, see Section 6.2)
aa	is the name of an additional attribute of the process object
+/-	is the sign + or -

The spaces between the elements are optional.

The event value is calculated by evaluating the expression defined by the formula and rounding the result to the nearest integer value.

The attribute has not been designed to shift the object value by a constant. For example, the formula 'OV + 1' is not valid. In such a case, use the VL (Value Low) attribute to achieve the desired result: Instead of defining formula 'OV + 1', simply set VL to -1.

Examples:

```
"10*OV"           ;The object value multiplied by ten
"2*OV + SB"       ;The OV and the attribute SB (Substituted)
"AS"              ;combined
"--9*OV/100 - 5*LI/9" ;The alarm state attribute is used as the object
                       ;value
                       ;for the event handling (instead of OV)
                       ;Valid formula which hardly makes sense
```

6.3.10 VL Value Low

The lowest of the distinct values that the process object state may have.

Data type:	Integer
Value:	Any integer value
Default value:	0
Access:	Read-only, configurable

When the VC (Value Count) attribute is given as a vector, this attribute has no meaning, see above.

For AEC event handling objects, the value is always 0.

6.3.11 ZT Modification Time

The time when the object was created or modified.

Data type:	Time
Access	Read-only

This attribute is set by the base system when the object is created, and it is updated each time the object is updated by the #MODIFY command (for example, by the event handling definition tool).

6.4 OPC Alarms & Events server attributes

6.4.1 OC OPC Condition Event

Specifies the CONDITION events that are generated.

Data type:	List	
Value:	The following attributes:	
	CD Text, up to 255 characters	Category description, the description of the category specified by CI
	CI Integer	Category ID, the OPC event category this event belongs to
	CN Text, up to 255 characters	The unique name of the condition
	SE Integer 0 ... 1000	The severity of the message generated when the condition is deactivated, 0 = no deactivation message sent
	SI List	Simple condition description, see below

Table continues on next page

SU Vector of list elements Subcondition descriptions of a multiple state condition, see below

Default: Empty list

Access: No restrictions

There may be several event handling objects that generate events of the same event category. In this case, the category description (CD) is typically defined in only one of them. If the description is given in several event handling objects, the descriptions must be identical.

A simple condition is described by the following subattributes of SI:

Data type:	List	
Value:	The following attributes:	
	CD Text, up to 255 characters	The definition of the condition
	SE Integer 0 ... 1000	The severity of the message generated when the condition is activated, 0 = no activation message sent

The subconditions of a multiple state condition are described by the elements of vector SU. Each element is a list with the following attributes:

Data type:	List	
Value:	The following attributes:	
	SD Text, up to 255 characters	The definition of the subcondition
	SE Integer 0 ... 1000	The severity of the message generated when the subcondition becomes active, 0 = no activation message sent
	SN Text, up to 255 characters	The name of the subcondition

The attributes SI and SU are mutually exclusive.

Multiple state conditions are applicable only for analog input objects. The value of the AZ (Alarm Zone) attribute of the process object acts as the index to the SU vector.

6.4.2 OG OPC Event Generation

Specifies which events (SIMPLE/TRACKING and/or CONDITION) are generated.

Data type:	Text keyword	
Value:	One of following:	
	"NONE"	No OPC events generated
	"SIMPLE_OR_TRACKING"	Simple or tracking event is generated
	"CONDITION"	Condition event is generated
	"BOTH"	Both events are generated
Default value:	"NONE"	
Access:	No restrictions	

6.4.3 OS OPC Simple Event

Specifies the SIMPLE or TRACKING events that are generated.

Data type:	List	
Value:	The following attributes:	
	CD	Text, up to 255 characters
	CI	Category description, the description of the category specified by CI
	ET	Category ID, the OPC event category this event belongs to
	SE	Event type to be generated, either "SIMPLE" or "TRACKING"
	SE	Severity (0 ... 1000) of the event, see below
Default value:	Empty list	
Access:	No restrictions	

There may be several event handling objects that generate events of the same event category. In this case, the category description (CD) is typically defined in only one of them. If the description is given in several event handling objects, the descriptions must be identical.

The severity (SE) of the event is an integer value 1 to 1000 to describe the urgency of the event (1 = lowest, 1000 = highest). SE value 0 is here used to state that no event notification is to be sent. If the SE is defined as a single integer value, this severity is used in all event notification messages regardless of the value (state) of the process object. If different severities for different events are required, these are given as an integer vector of length VC (Value Count, see [Section 6.4](#)). The severities in the vector correspond to the distinct values of the new state of the object.

6.5 OPC Alarms & Events client attributes

6.5.1 CD Event Category Description

The description of the event category specified by the CI attribute.

Data type:	Text
Value:	Any text, up to 255 characters
Default value:	""
Access:	Read-only, configurable

This attribute is used only for documentation purposes.

6.5.2 CI Event Category ID

The OPC event category ID.

Data type:	Integer
Value:	Server specific
Default value:	-1
Access:	Read-only, configurable

This attribute defines the event category whose SIMPLE or TRACKING events (see the ET attribute) the process objects will receive. Special value -1 specifies that the process objects of this event handling receive events from any category.

For CONDITION events, the attribute has only a documentation meaning, because the condition names are unique within a server.

6.5.3 CN Condition Name

The name of the condition that the event handling object is for.

Data type:	Text
Value:	Server specific text, up to 255 characters
Default value:	""
Access:	Read-only, configurable

This attribute is relevant only for CONDITION events.

6.5.4 EM Event Messages

List of event messages of the SIMPLE or TRACKING event.

Data type:	Text vector
Value:	Up to 255 server specific event messages
Default value:	Empty vector
Access:	Read-only, configurable

This attribute is used to translate a SIMPLE or TRACKING event to a numeric value of the process object (attribute OE). If the event message matches the first element of the vector, OE is set to 0. If it matches the second element, OE is set to 1, etc. The last element may be set to "*" to match with any message not mentioned in the list before. If no match is found, the OE is set to the length of the vector.

6.5.5 ET Event Type

The OPC event type handled by this event handling object.

Data type:	Text keyword
Value:	The OPC event type:
	"SIMPLE"
	"TRACKING"
	"CONDITION"
Access:	Read-only, configurable

For SIMPLE and TRACKING events, attributes ND, CI and ET uniquely define the events that the process objects of this event handling object will receive.

For CONDITION events, the received events are uniquely defined by attributes ND, ET and CN.

6.5.6 ND Node

The number of the node object that specifies the OPC A&E Server in question.

Data type:	Integer
Value:	1 ... 250, the node number
Access:	Read-only, configurable

Note: The AEC event handling objects are OPC A&E Server specific. Consequently, each server used by the application has its own set of event handling objects.

6.5.7 SN Subcondition Names

List of sub-condition names of the CONDITION event.

Data type:	Text vector
Value:	Up to 255 server specific sub-condition names
Default value:	Empty vector (SIMPLE and TRACKING events)
	One-element vector containing the name of the CONDITION
Access:	Read-only, configurable

This attribute is used to translate a condition event to a numeric value of the process object (attribute OE). If the condition is inactive, the value is 0. If the condition is active, the numeric value is the ordinal number of the sub-condition name within the list in the SN attribute.

According to the OPC standard, the only sub-condition of a simple condition has the name of the condition itself (a condition is simple when it has only two states). Hence, the default value of this attribute contains the name of the condition.

6.6 Defining event handling objects using SCIL

Example:

Creating an event handling object called MY_EVENT:

```
#CREATE MY_EVENT:H = LIST(VC = 2, VL = 0,-
    ST = ("TEXT_Open_State", "TEXT_Closed_State"),-
    MT = ("TEXT_Still_Open", "TEXT_Closed", "TEXT_Opened",
    "TEXT_Still_Closed"))
```

Note that different text identifiers are used for the state "closed" and for the event "closed". In many languages, distinct words are used for the two.

6.7 Predefined event handling objects

Two types of event handling objects are predefined by the base system:

- Generic object value related event handling objects for various types of process objects. These objects are applied to process object events when no application specific event handling is defined (the EH attribute of the process object is not set).
- Generic event handling objects for events caused by the changes of non-OV attributes.

The predefined event handling objects may not be modified or deleted. The English translations of the texts are found in SYS_TEXT.SDB shipped along with the base system. This file should not be modified. Site-specific changes, as well as translations into other languages should be included in an application defined text database.

6.7.1 OV related predefined event handling objects

[Table 5](#) below lists the OV related predefined event handling objects and the process object types they are applied on.

The text identifiers in the ST (State Text) attribute are named as "object_ST_n", where "object" is the name of the H object and "n" is the state. For example, SYS_BI_ST_1 refers to the state text for the BI value 1 of a non-alarming binary input object.

The text identifiers in the MT (Message Text) attribute are named as "object_MT_p_n", where "object" is the name of the H object, "p" is the previous state and "n" is the new state. For example, SYS_BI_MT_0_1 refers to the message text for the event caused by the BI value change from 0 to 1.

Table 5: The OV related predefined event handling objects

The name of the H object	Process object types
SYS_BI	BI objects, AC = 0
SYS_BI_AGO	BI objects, AG = 0
SYS_BI_AG1	BI objects, AG = 1
SYS_BI_AG2_NV0	BI objects, AG = 2, NV = 0
SYS_BI_AG2_NV1	BI objects, AG = 2, NV = 1
SYS_BI_AG2_NV2	BI objects, AG = 2, NV = 2
SYS_BI_AG3	BI objects, AG = 3
SYS_BI_AG4	BI objects, AG = 4
SYS_BI_AG5	BI objects, AG = 5
SYS_BO	BO objects
SYS_DI	DI objects
SYS_DO	DO objects
SYS_DB	DB objects, AC = 0
SYS_DB_LA0	DB objects, LA = [0]
SYS_DB_LA1	DB objects, LA = [1]
SYS_DB_LA2	DB objects, LA = [2]
SYS_DB_LA3	DB objects, LA = [3]
SYS_DB_LA01	DB objects, LA = [0,1]
SYS_DB_LA02	DB objects, LA = [0,2]
SYS_DB_LA03	DB objects, LA = [0,3]
SYS_DB_LA12	DB objects, LA = [1,2]
SYS_DB_LA13	DB objects, LA = [1,3]
SYS_DB_LA23	DB objects, LA = [2,3]
SYS_DB_LA012	DB objects, LA = [0,1,2]
SYS_DB_LA013	DB objects, LA = [0,1,3]
SYS_DB_LA023	DB objects, LA = [0,2,3]
Table continues on next page	

The name of the H object	Process object types
SYS_DB_LA123	DB objects, LA = [1,2,3]
SYS_DB_LA0123_NV0	DB objects, LA = [0,1,2,3], NV = 0
SYS_DB_LA0123_NV1	DB objects, LA = [0,1,2,3], NV = 1
SYS_DB_LA0123_NV2	DB objects, LA = [0,1,2,3], NV = 2
SYS_DB_LA0123_NV3	DB objects, LA = [0,1,2,3], NV = 3
SYS_DB_LA0123_NV4	DB objects, LA = [0,1,2,3], NV = 4
SYS_AI	AI objects, AC = 0
SYS_AI_AZ2	AI objects, two alarm limits (LI, HI)
SYS_AI_AZ4	AI objects, four alarm limits (LI, LW, HW, HI)
SYS_AO	AO objects, AC = 0
SYS_AO_AZ2	AO objects, AC <> 0
SYS_PC	PC objects
SYS_BS	BS objects
SYS_FT	FT objects
SYS_NT_POWER	NT objects of the POWER schema

6.7.2

Predefined event handling objects for non-OV events

[Table 6](#) below lists the predefined event handling objects that are applied for events caused by changes of non-OV attributes.

The text identifiers in the ST (State Text) attribute are named as "object_ST_n", where "object" is the name of the H object and "n" is the state. For example, SYS_AB_ST_1 refers to the state text for the "alarm blocked" state of the object.

No message texts are defined for these event handling objects, the state texts are used as message texts as well.

Table 6: The predefined event handling objects for non-OV events

The name of the H object	Changed attribute	States
SYS_AB	AB	0 ... 1
SYS_AC	AC	0 ... 7
SYS_AG	AG	0 ... 5
SYS_AR	AR	0 ... 1
SYS_BL	BL	0 ... 1
SYS_HB	HB	0 ... 1
SYS_HI	HI	None
SYS_HO	HO	None
SYS_HW	HW	None
SYS_IU	IU	0 ... 1
SYS_LA	LA	0 ... 15
SYS_LI	LI	None
SYS_LO	LO	None
SYS_LW	LW	None
SYS_OF	OF	0 ... 1

Table continues on next page

The name of the H object	Changed attribute	States
SYS_OR	OR	0 ... 1
SYS_OS	OS	0 ... 10
SYS_PB	PB	0 ... 1
SYS_SB	SB	0 ... 1
SYS_SE	SE	0 ... 1
SYS_SP	SP	0 ... 1
SYS_SS	SS	0 ... 3
SYS_SU	SU	0 ... 1
SYS_UB	UB	0 ... 1
SYS_XB	XB	0 ... 1

6.8 Application default event handling objects

If the predefined event handling objects described above are not adequate for the application, they can be overridden by application default event handling objects. The most common need to modify the predefined event handling is to add OPC event generation functionality to it (attributes OS, OC and OG).

The names of application default event handling objects are fixed: they are formed by replacing the "SYS" prefix of the corresponding predefined event handling object by "APL". For example, object APL_BI is the application default event handling object for Binary Input objects whose alarm class is 0.

It is recommended that application default event handling objects are created by first copying the corresponding predefined event handling object and then adding new functionality to it.

Section 7 Scales

7.1 About this section

This section describes scale objects and their attributes:

- [Section 7.2](#) [General](#): The basic properties of scale objects, their use and function, etc.
- [Section 7.3](#) [Scale attributes](#): The Scale attributes listed and described in alphabetical order.
- [Section 7.4](#) [Defining scale objects using SCIL](#): Required attributes and an example.

7.2 General

7.2.1 Use

Scales define algorithms for scaling the analog data received from the process stations to the unit used in the process database.

Every definition of an analog process object contains a scale name (the process object attribute SN), which is the name of the scale to be used in the scaling. The same scale can be used by several analog process objects, independent of their type (analog input or output).

7.2.2 Function

The stations (RTUs, protective equipment, PLCs and other process control units) receive analog signals from the process instrumentation as electrical currents (mA) or voltages (V). In the stations, these values are converted to digital values. In the process database the digital values are scaled to the analog unit of the process object using the scaling algorithm given by the scale name of the process object. The analog output values are scaled correspondingly in the process database before they are sent out to the stations.

Three scaling algorithms are supported:

1. One-to-one scaling (no scaling)
2. Linear scaling
3. Stepwise linear scaling, this algorithm approximates a non-linear scaling curve by a broken line.

As an example, scaling can mean that digital value 0 corresponds to 20 °C and value 1000 corresponds to 100 °C (see [Figure 9](#)).

7.2.3 Storage

The maximum number of scale objects in an application is 65 535. Scale objects are stored in the process database file APL_PROCES.PRD. When the application is WARM or HOT, the scale objects are stored in the global memory pool of SYS600 as well (see the System Objects manual).

7.2.4 Scale object notation

Scale objects and their attributes are accessed from SCIL with the following object notation (see also [Section 4](#)).

name:{application}X{attribute}{(index)}

or

name:{application}X{attribute}{index}

where

'name'	is the name of the object
'application'	is the logical application number
'attribute'	is the attribute name
'index'	is an index or index range

7.3 Scale attributes

7.3.1 LN Logical Name

The name of the scale.

Data type:	Text
Value:	Object name
Access	Read-only, configurable

7.3.2 SA Scaling Algorithm

The algorithm used in scaling: one-to-one, linear or stepwise linear scaling, see [Figure 9](#).

Data type:	Integer
Value:	0 1:1 scaling
	1 Linear scaling
	2 Stepwise linear scaling
Default:	0
Access:	Read-only, configurable

7.3.3 SC Scaling Constants

Constants that specify the scaling curve. The curve is given as pairs of corresponding station values and SYS600 process database values.

Data type:	Vector
Value:	If SA = 1: Four real elements If SA = 2: Up to 100 real elements representing the coordinates of 50 points on the scaling curve
Indexing:	If SA = 1:

Table continues on next page

	Index 1	The lower station value
	Index 2	The upper station value
	Index 3	The lower process database value
	Index 4	The upper process database value
	If SA = 2:	An odd index refers to a station value and the following even index to corresponding process database value. The station values must be given in ascending order.
Default value:	No	
Access:	Read-only, configurable	

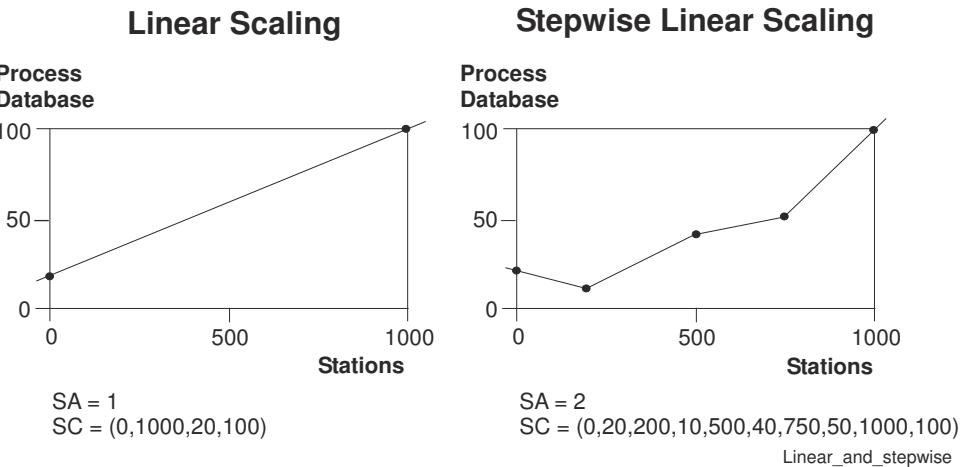


Figure 9: An illustration of linear and stepwise linear scaling

7.3.4 ZT Modification Time

The time when the object was created or modified. This attribute is set by the base system when the scale is created, and it is updated each time the scale is updated by the #MODIFY command (for example, by the scale definition tool).

Data type:	Time
Access:	Read-only

7.4 Defining scale objects using SCIL

7.4.1 Required attributes

When defining a new scale object, SA is the only mandatory attribute. If the SA attribute is set to 1 or 2, the SC attribute must be defined as well.

Example:

Creating a one-to-one scale:

```
#CREATE ONE_TO_ONE:X = LIST(SA = 0)
```

Creating a linear scale named LINEAR:

```
#CREATE LINEAR:X = LIST(SA = 1, SC = (1,100,10,500))
```


Section 8 Data objects

This section describes the data objects, their attributes and how to define data objects with SCIL. It is divided into the following sections:

- [Section 8.1](#) **General:** The use, activation and function of the data objects, etc.
- [Section 8.2](#) **Data object attributes:** The data object attributes are listed and described. The attributes are grouped in sub-sections according to their function.
- [Section 8.3](#) The principles for defining data objects using SCIL and examples.

8.1 General

8.1.1 Use

Data objects (also called datalog objects) are used to sample and calculate, register and store data. A data object can contain one or more, up to 2 000 000, registered values. Each value has a validity stamp (status code) and a time stamp.

Data objects can be used for storing trend data, historical data, running plan data, data for system configuration, optimization, calculation, etc. Data objects can also be used as application wide variables when there is a need for using the same data in several different SCIL contexts.

8.1.2 Function

Execution of a data object is carried out in the following steps:

1. The **raw value** of the object is first obtained by evaluating a user defined SCIL expression (attribute IN). Typically, the expression contains a reference to another application object, such as "VOLTAGE:PAI5". However, any valid SCIL expression resulting in a proper data type will do. The expression may also refer to the argument variables supplied by the activator of the object, see [Section 8.1.3](#). The data type of the object is specified by attribute VT. Data types REAL, INTEGER, TEXT and TIME are supported.
2. A **logging function** (attribute LF) is applied to the raw value, resulting to the **object value** (attribute OV). The simplest logging function, DIRECT, just sets OV equal to the raw value, others perform various time series calculations, such as averaging, accumulation, integration, slope calculation and minimum/maximum search, on the raw value. There is also a logging function that makes a copy of the contents of another data object, bypassing step 1 altogether.
3. The new object value is time-stamped and stored in the object as attributes OV, OS, QT and RT. The new values are written to the disk, unless the object is defined as "memory only" (attribute MO).
4. If history registration is defined for the object (attribute HR > 0), the new object attribute values OV, OS and RT are also stored in the history of the object. The history of the object is arranged as a cycling buffer: when the maximum number of history entries has been reached, the oldest entries are lost. The history may also be reset, see initialization below.

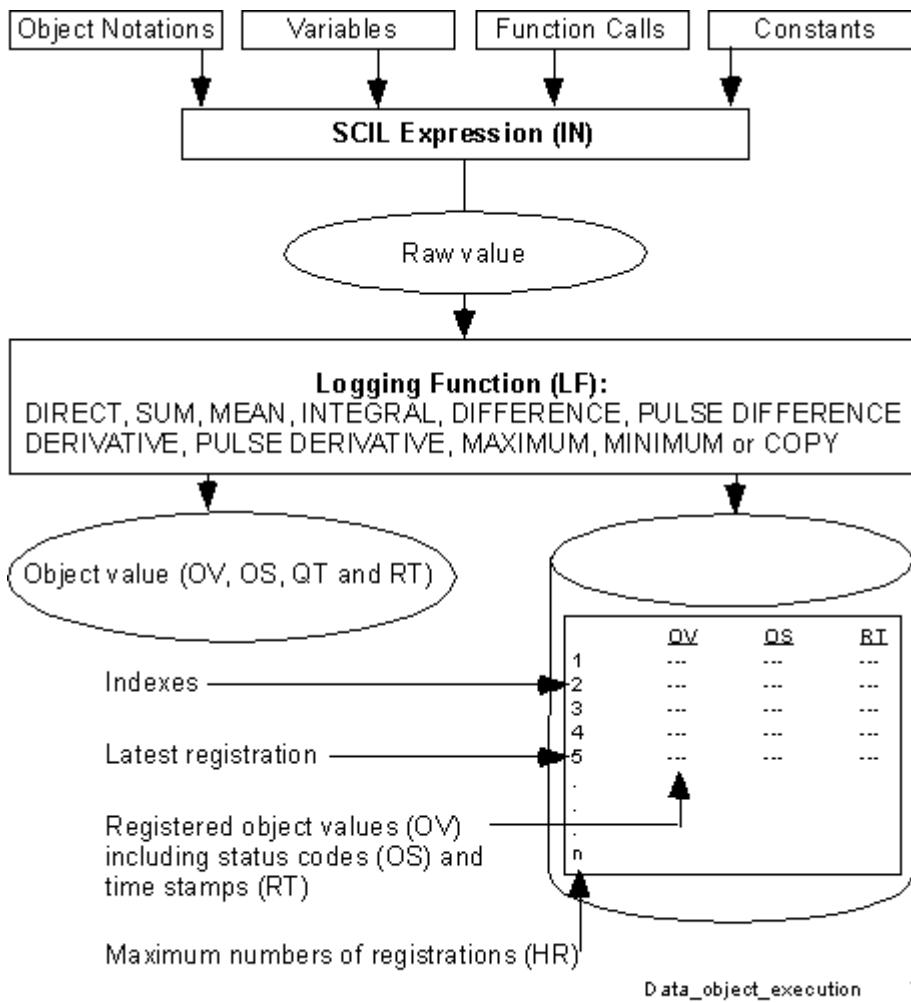


Figure 10: Execution of a data object

Data objects connected to a time channel are **initialized** by the time channel when it is scheduled for its initialization run (see [Section 10](#)). Initialization is carried out in the following steps:

1. A new time series calculation is started. This applies to logging functions SUM, MEAN, INTEGRAL, MAXIMUM and MINIMUM (see attribute LF).
2. The history of the object (if any) is cleared, the number of history registrations (attribute LR) is set to zero.

A data object is activated in the following three ways:

- By a SCIL program with commands #EXEC and #EXEC_AFTER (see [Section 4](#)), for example, #EXEC DATA:D.
- By a time channel ([Section 10](#)), giving an automatic time based execution.
- By an event channel ([Section 11](#)), giving an automatic event based execution triggered by a process or a system event.

It is possible to use all the three ways for activating the same data object.

8.1.3 Variables

The following variables may be used in the SCIL expression of the IN attribute:

- argument variables given as a variable list of #EXEC or #EXEC_AFTER command, when activated by SCIL (see [Section 4](#)).
- argument variables of the event channel, when activated by an event channel (see [Section 11](#)).
- variables of the time channel, when activated by a time channel. These are the variables created by the command procedures already executed by the time channel.

8.1.4 Accessing registered data with SCIL

The registered data, that is, attributes OV, OS, RT and QT, may be accessed in two ways by SCIL, with or without indexing. The newest value is always referred to when indexing is not used. The history registrations (if any) are accessed by indexing, the oldest entry has index 1 and the latest has the index specified by the LR attribute of the object. If LR = 0, there are no history entries.

The registered data may be changed using #SET command. This is frequently done when the data object is used as an application wide storage with no automatic data logging. Also, values that are missing or known to be incorrect may be manually entered by the operator to complete the reporting. When patching incorrect values, read attributes OS and RT first, then set the new OV value and rewrite OS and RT. This is because setting OV clears OS and updates RT.

History registrations may also be read and written with SCIL functions DATA_FETCH and DATA_STORE, see Programming Language SCIL manual.

8.1.5 Execution queues

Data objects, as well as command procedures, time channels and event channels, are executed by **execution queues** of the report database. An execution queue may be seen as a queue of execution requests and an active base system program that executes these requests. The program is implemented as a separate process within the underlying operating system. Therefore, the queues are executed in parallel.

There are three types of execution queues:

1. The **time channel queue** schedules and executes the time channels. The request queue is organized as a time schedule. When a time channel has been executed, it is rescheduled according to its scheduling attributes.
2. The **event channel queue** is a multi-purpose queue that executes all the other (non-parallel, see below) execution requests. As its name implies, its most important task is to execute the event channels triggered by various process and system events. It also executes the requests done by SCIL in human interface programs. The request queue is organized as a FIFO (first in first out) queue.
3. **Parallel queues** are application defined queues that are normally used to optimize the throughput of the system. There may be 0 to 30 parallel queues in an application, the number is specified by the base system object attribute APL:BPQ. The mutual priority of the queues (in the sense of CPU usage) may be defined (attribute APL:BQP). A parallel queue may be dedicated or non-dedicated (attribute APL:BQD). A dedicated queue accepts only requests that are explicitly given to it, a non-dedicated queue accepts also parallel requests with non-specified queue number. The request queue is organised as a FIFO.

A data object and a command procedure object is always executed by the time channel queue or the event channel queue, unless it has specified parallel execution (attribute PE = 1) for itself. The parallel queue to execute the object may be explicitly specified (attribute PQ > 0) or it may be left unspecified (PQ = 0). If unspecified, the system will assign the first idle queue to it, see [Section 8.2.4](#).

The objects activated by a command procedure object using #EXEC command are executed in the same queue where the command procedure is running.

The queue that will execute the object may be determined by [Figure 11](#).

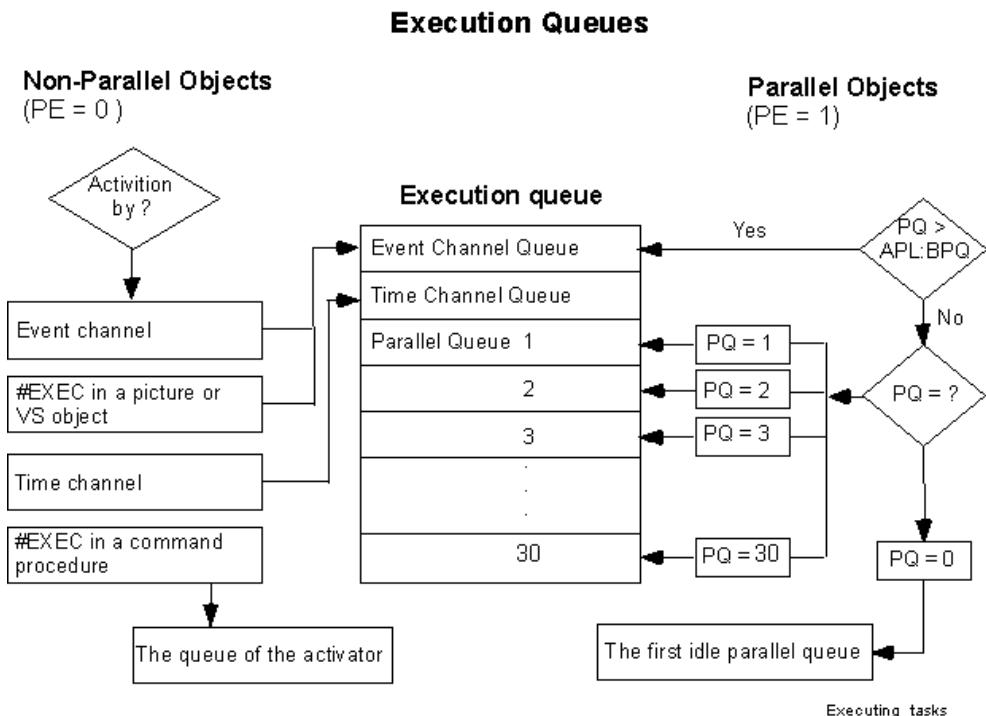


Figure 11: The queues that execute data objects and command procedures

In addition to the execution queues described above, there are two virtual execution queues that only hold requests until they are moved into a true execution queue:

1. **Delayed execution queue** holds requests until the delay expires. These are the requests activated by #EXEC_AFTER command of SCIL language.
2. **Parallel queue 0** holds parallel requests with PQ = 0 until a (non-dedicated) parallel queue becomes idle.

The contents of the queues may be examined in SCIL programs by reading various application object attributes (see the System Objects manual for details):

- Attributes EU (Event Channel Queue Used) and QU (Queue Used) give information about the lengths of queues
- Attribute RO (Running Objects) lists the objects currently in execution
- Attribute QO (Queued Objects) lists the objects that are queued but not yet executing

8.1.6 Storage

The combined maximum number of data and command procedure objects in an application is 2 000 000. Data objects are stored in the report database files APL_REPORT.000 to APL_REPORT.399. When the application is WARM or HOT, the data objects along with most of their attributes are stored in the global memory pool of SYS600 as well (see the System Objects manual).

The history registrations (attributes OV, OS, QT and RT, indexed values) are not permanently stored in the global memory pool. Instead, the most recently used history data is cached in the report cache along with the programs of command procedure objects. The memory space for

the report cache is allocated from the global memory pool (see the System Objects manual, base system object attribute RC).

For optimization of the throughput, the dynamically changing attributes OV, OS, QT and RT may be defined to be "memory only", see attribute MO. In this case, they are not stored in the disk file at all.

8.1.7 Data object notation

Data objects and their attributes are accessed from SCIL with the following object notation (see also [Section 4](#)):

name:[application]D{attribute}{(index)}

or

name:[application]D{attribute}{index}

where

'name' is the name of the object

'application' is the logical application number

'attribute' is the attribute name

'index' is an index or index range

Depending on the context, an object notation without an attribute refers to the entire object, for example in command #EXEC, or to the default attribute OV, for example in #SET command and as an operand of an expression.

Indexing is used to refer to the history registrations of the object (attributes OV, OS, QT and RT).

8.2 Data object attributes

In this section, the data object attributes are grouped into the following sub-sections:

Section 8.2.1	Basic attributes:	CM, FI, FX, IU, LN, ON
Section 8.2.2	Logging attributes:	HR, IN, LF, PS, SR, TS, VL, VT
Section 8.2.3	Historian logging:	GN, GP
Section 8.2.4	Execution control:	EP, PE, PQ, SE, TC
Section 8.2.5	Storage:	HN, MO
Section 8.2.6	Registered data:	LR, OS, OV, QT, RT

The data attributes in [Section 8.2.6](#) represent the registered data. All other attributes describe the definition of the object.

8.2.1 Basic attributes

8.2.1.1 CM Comment

A freely chosen comment text.

Data type:	Text
Value:	Any Unicode text, up to 255 characters
Access:	No restrictions

8.2.1.2 FI Free Integer

This attribute is reserved for the SCIL application. It has no functionality in the base system.

Data type:	Integer
Default value:	0
Access:	No restrictions

8.2.1.3 FX Free Text

This attribute is reserved for the SCIL application. It has no functionality in the base system.

Data type:	Text
Value:	Any Unicode text, up to 255 characters
Default value:	""
Access:	No restrictions

8.2.1.4 IU In Use

The attribute states whether the object is in use or not. When the object is out of use ($IU = 0$), it is not executed. However, the attributes can be both read and written normally.

Data type:	Integer	
Value:	0	Out of use
	1	In use
Default value:	0	
Access:	No restrictions	

8.2.1.5 LN Logical Name

The logical name of the data object.

Data type:	Text
Value:	Object name
Access:	Read-only, configurable

8.2.1.6 ON OPC Item Name

The alias name of the object to be used as an OPC item id.

Data type:	Text
Value:	Up to 255 Unicode characters. For the syntax, see below.
Default value:	""
Access:	No restrictions

The value of the attribute is a hierarchical name that consists of any number of fields separated by periods:

"field1.field2.fieldn"

Each field may contain any visible characters, except for colons and periods. The name is case-sensitive. Single embedded spaces are allowed in the field, whereas leading and trailing spaces are not accepted.

Examples of the valid ON attribute values:

- "South Tipperary.Kilkenny.Relay1.Breaker.Position"
- "Äänekosken ala-asema.Laukaan syöttö.Katkaisija.Tila"

The names are unique within an application. Consequently, the user cannot, for example, give the same ON attribute value to a process object and a data object.

The hierarchy formed by the ON attribute values is shown by the name space browser of the OPC Server.

8.2.1.7 ZT Modification Time

The time when the object was created or modified. This attribute is set by the base system when the object is created, and it is updated each time the object is updated by the #MODIFY command (for example, by the data object definition tool).

Data type:	Time
Access:	Read-only

8.2.2 Logging attributes

8.2.2.1 HR History Registrations

The maximum number of history registrations. When the number of registrations (attribute LR) exceeds this value, the oldest value is lost.

Data type:	Integer
Value:	0 ... 2 000 000
Default value:	0
Access:	Read-only, configurable

When a new data object is created or HR attribute is modified, the disk or memory space required to store the history data is allocated from the report database file or from the global memory pool, depending on the MO (Memory Only) attribute value. When HR is lowered below the current number of registrations (attribute LR), the oldest values are lost and LR is set equal to HR. When HR is raised, the old values are preserved.

Because the newest value of the object is always available in attributes OV, OS, QT and RT (without indexing), it normally makes no sense to set HR to 1. When HR is 1, OV and OV(1) are always equal except for the time interval between the initialization and next execution of the object. Within this interval, LR is 0 and OV(1) does not exist.

8.2.2.2 IN Instruction

The SCIL expression to evaluate the raw value of the object.

Data type:	Text
Value:	SCIL expression, up to 255 characters
Access:	Read-only, configurable

The evaluation of this attribute must result in the data type defined by the VT attribute. However, if VT is "INTEGER", the following data conversions are done automatically: real values are rounded to nearest integer and boolean values are converted to 0 (= FALSE) or 1 (= TRUE).

8.2.2.3 LF Logging Function

The operation to be performed on the raw value to calculate the object value OV. The previous value of OV and/or the previous raw value is used by many logging functions to calculate the new object value (see [Figure 10](#)). The allowed value types (attribute VT) for each logging function are given in the table below.

Data type:	Integer
Value:	0 ... 10:
0	DIRECT. No calculation is performed. The object value is the same as the raw value. VT: ALL
1	SUM. The cumulative sum of all raw values since last initialization. VT: REAL, INTEGER
2	MEAN. The mean (or average) value of all raw values since last initialization. VT: REAL
3	INTEGRAL. The time integral of the raw value since last initialization. The raw value is assumed to be constant between samplings. The time is measured in seconds. The object value is calculated by adding the product of the previous raw value and the elapsed time to the previous object value. VT: REAL
4	DIFFERENCE. The difference between two consecutive raw values. The object value is calculated by subtracting the previous raw value from the current. VT: REAL, INTEGER
5	PULSE DIFFERENCE. The same as above, but the raw value is regarded to be a pulse counter, which is reset to zero at certain value (attribute PS). VT: REAL, INTEGER
6	TIME DERIVATIVE. The time derivative (slope) of two consecutive raw values. The time is measured in seconds. The object value is calculated by dividing the difference of the current and previous raw value by the elapsed time. VT: REAL
7	PULSE DERIVATIVE. The same as above, but the raw value is regarded to be a pulse counter, which is reset to zero at certain value (attribute PS). VT: REAL
8	MAXIMUM. The object value is the largest of the raw values since last initialization. VT: REAL, INTEGER, TIME
9	MINIMUM. The object value is the smallest of the raw values since last initialization. VT: REAL, INTEGER, TIME
10	COPY. Copies all the registered values from another data object. (specified by SR attribute). IN attribute has no meaning. VT: ALL
Default:	0 (DIRECT)
Access:	Read-only, configurable

How to use different logging functions:

- INTEGRAL gives the energy if the raw value measures power.
- PULSE DIFFERENCE gives the energy of last cycle if the raw value is a pulse counter measuring energy.
- PULSE DERIVATIVE gives the mean power of last cycle if the raw value is a pulse counter measuring energy.
- COPY is used to save the contents of a data object with a different name, for example to save, at midnight, the hourly values of the day as "yesterday's" values.



When the value of this attribute is changed, all the history registrations are lost and LR is set to 0.

8.2.2.4 PS Pulse Scale

When the logging function is PULSE DIFFERENCE (LF = 5) or PULSE DERIVATIVE (LF = 7), the raw value (result of evaluating the IN attribute) is regarded as a pulse counter. The PS attribute indicates the width of the pulse counter, that is, the value that resets the physical pulse counter to zero. In other words, the PS attribute should be set to the maximum value of the pulse counter + the size of one pulse. For other logging functions, this attribute has no meaning.

Data type:	Integer or real (according to VT attribute)
Value:	Non-negative number
Default value:	0
Access:	Read-only, configurable

8.2.2.5 SR Source

The name of the data object to be copied when the logging function is COPY (LF = 10). For other logging functions, this attribute has no meaning and is set to an empty string.

Data type:	Text
Value:	Object name
Default value:	""
Access:	Read-only, configurable

8.2.2.6 TS Time Stamp

The TS attribute specifies whether the attributes RT and QT get their value from the system clock or from the (argument) variables QT, RT and RM.

Data type:	Integer
Value:	0 RT and QT are read from the system clock
	1 RT and QT are read from variables QT, RT and RM
Default:	0
Access:	No restrictions

When TS = 1, the registration time of the object is set according to the following rules:

1. If argument variable QT is given and it contains a valid qualified time, it is used as the registration time.
2. Otherwise, if argument variable RT is given and it contains a time value, it is used as the registration time (possibly further specified by argument variable RM).
3. If neither a valid QT nor RT argument variable is given, the registration time is read from the system clock.

When the object is started by a process event channel or by a predefined event channel, argument variables QT, RT and RM normally contain the time of the triggering event. When TS attribute is set to 1, the RT and QT attributes of the data object will match the time of the event.

8.2.2.7 VL Value Length

Maximum value length. This attribute specifies the maximum length of the text type OV attribute. If the value of VT attribute is not "TEXT", this attribute has no meaning, and it is set to 0.

Data type:	Integer
Value:	0 ... 255
Access:	Read only, configurable



When the value of this attribute is changed, all the history registrations are lost and LR is set to 0.

8.2.2.8 VT Value Type

The value type of the logged data (attribute OV).

Data type:	Text keyword
Value:	"REAL", "INTEGER", "TIME" or "TEXT"
Default value:	"REAL"
Access:	Read only, configurable with #CREATE but not with #MODIFY.

8.2.3 Historian logging

The Historian logging attributes specify how the changes of the OV (Object Value) attribute are stored in Historian database(s). Only data objects of value type REAL or INTEGER may be logged.

For more information about Historian logging, see [Section 12](#).

8.2.3.1 GN Logging Name

This attribute tells the names of Historian database tags that receive data from this data object.

Data type:	Vector
Element type:	List
Element value:	DB The name of the DATABASE type logging profile object that defines the Historian database

Table continues on next page

	GN	The name of the tag in this Historian database, see below
Default:	Empty vector	
Access:	Read-only	

The tags are finally named by the Historian database itself. When the tag is not yet created, the GN attribute returns the proposed tag name in square brackets.

8.2.3.2 GP Logging Profile

This attribute specifies the logging profile to be used when logging values to the SYS600 Historian database(s).

Data type:	Text
Value:	Name of an OBJECT type logging profile object. If empty, no logging is done.
Default value:	""
Access:	No restrictions

8.2.4 Execution control

8.2.4.1 EP Execution Priority

The execution order of the object relative to other data objects and command procedures within the same time channel. Objects with the same EP value may be executed in any order.

Data type:	Integer
Value:	0 ... 255
	0 is the highest and 255 the lowest priority order
Default value:	255
Access:	Read-only, configurable

8.2.4.2 PE Parallel Execution

This attribute indicates whether the object is executed by a parallel queue or not, see [Figure 11](#).

Data type:	Integer	
Value:	0	Non-parallel execution
	1	Parallel execution.
Default value:	0	
Access:	No restrictions	

8.2.4.3 PQ Parallel Queue

The number of the parallel queue that will execute the object. If PE = 0, this attribute has no meaning.

Data type:	Integer	
Value:	0 ... APL:BPQ	
Value 0 means that any parallel queue will do.		
Default value:	0	
Access:	No restrictions	

8.2.4.4 SE Start-up Execution

This attribute indicates whether the data object is executed during application start-up, when reporting is chasing the real time. If start-up execution is specified, the history registrations that have been missed during the application shutdown are time-stamped with the scheduled time and marked with NOT_SAMPLED_STATUS (OS = 10).

Data type:	Integer	
Value:	0	No
	1	Yes
Default value:	0	
Access:	No restrictions	

8.2.4.5 TC Time Channel

The name of the time channel that runs the data object.

Data type:	Text	
Value:	Object name	
Access:	Read-only, configurable	

8.2.5 Storage

8.2.5.1 HN History File Number

Data objects are stored in files named APL_REPORT.nnn, where "nnn" is a three-digit sequence number. This file number is specified by the HN attribute. The file numbers can be freely chosen from the range 0 ... 399.

Data type:	Integer	
Value:	0 ... 399	
Default value:	0	
Access:	Read-only, configurable	

Note: Division of the report database into several files was, in the first place, implemented due to the 32 MB size limit of MicroSCADA files. Now that no such restriction exists anymore, using HN values other than 0 is more or less obsolete.

8.2.5.2 MO Memory Only

This attribute determines whether the dynamically changing attributes OV, OS, QT, RT and LR are stored on disk. If old values (stored before last start-up) of these attributes are not interesting to the application, some performance gain may be achieved by setting this

attribute to 1. Global memory pool must be defined large enough to hold all the history registrations of memory-only data objects.

Data type:	Integer
Value:	0 Dynamic attributes are stored on disk
	1 Dynamic attributes are stored only in RAM
Default value:	0
Access:	Read-only, configurable



When the value of this attribute is changed, all the history registrations are lost and LR is set to 0.

8.2.6 Registered data

8.2.6.1 LR Latest Registration

The index of the latest registered value, that is, the current number of history registrations. This attribute is automatically updated, but it may be changed by #SET command. If the data object is used as a data storage of a SCIL application (not connected to any time channel), LR should be set equal to HR, because only indices less or equal to LR can be read by SCIL. An index above LR (if not above HR) may, however, be written by #SET command. The change becomes visible only after LR has been raised.

Data type:	Integer
Value:	Less or equal to the attribute HR
Access:	No restrictions.

8.2.6.2 OS Object Status

SCIL status code that describes the reliability of the object value.

Data type:	Integer
Value:	A SCIL status code, see the manual Status Codes.
Indexing:	History registration number. Without indexing, the newest value.
Access:	No restrictions

The value of this attribute can be any status code produced by the evaluation of the IN attribute. In addition, the following status codes are used:

- NOT_SAMPLED_STATUS (10), used to indicate that the object has not been executed. This status is set for a new object and for each missed scheduled execution.
- SUSPICIOUS_STATUS (1), used to indicate that the object value may be unreliable. This status is set when the logging function is not DIRECT and evaluation of one or more raw values has failed. For example, when the logging function is SUM, it indicates that one or more of the values have a non-zero status or are totally missing from the sum. The raw value status FAULTY_TIME_STATUS (3) is considered here as a good status and does not set OS to SUSPICIOUS_STATUS.

8.2.6.3 OV Object Value

The object value.

Data type:	Real, integer, text or time. Defined by the VT attribute.
Indexing:	History registration number. Without indexing, the newest value.
Access:	No restrictions



When setting OV values with the #SET command, the newest value and the latest history value are independent. For example, if there are 10 history values stored and OV(10) is set, the "newest" value OV does not change. Similarly, setting OV does not change OV(10). Setting OV (with or without indexing) clears OS and updates RT and QT.

8.2.6.4 QT Qualified Registration Time

The registration time of the object value with a resolution of one millisecond. The time is either read from the system clock or received as argument variables QT, RT and RM, according to the TS attribute value.

Data type:	List
Value:	Qualified time, attributes
	CL Time
	MS Integer, milliseconds
	DS Boolean, daylight saving
Indexing:	History registration number. Without indexing, the newest value.
Access:	No restrictions



The milliseconds are not stored in the object history. Therefore, the MS attribute of the list value is always zero when indexing is used. However, the daylight saving flag (DS) is valid.

8.2.6.5 RT Registration Time

The registration time of the object value with a resolution of one second. The time is either read from the system clock or received as argument variable QT or RT, according to the TS attribute value.

Data type:	Time
Indexing:	History registration number. Without indexing, the newest value.
Access:	No restrictions.

8.3 Defining data objects using SCIL

Only the name (LN attribute) is required when a new data object is created. Other attributes get the default values mentioned in the attribute descriptions, for example:

Expression, IN	No expression
Logging function, LF	DIRECT
Number of history registrations, HR	0
Execution priority, EP	255
In use, IU	0

When a new data object is created, its OS (Object Status) attribute is set to NOT_SAMPLED_STATUS (= 10) and LR (Latest Registration) to 0.

Examples:

In the example below, a data object named "DATA" is created with the following attributes:

Expression	%A
Logging function	PULSE DIFFERENCE
Maximum history registrations	2000
State of use	In use

```
#CREATE DATA:D = LIST(IN = "%A", LF = 5, HR = 2000, IU = 1)
```

In the following example, an existing data object is copied using the FETCH function:

```
V = FETCH(0, "D", "DATA1")
#CREATE DATA2:D = V
```

Only the definition of object "DATA1" is copied by this example, not the registered data.

Section 9 Command procedures

This section describes the command procedures and their attributes. The section is divided into the following sections:

- [Section 9.1](#) **General**: The use of command procedures, the activation and function of command procedures, the use of variables in command procedures, etc.
- [Section 9.2](#) **Command procedure attributes**: The attributes listed and described. The attributes are grouped into sub-sections according to their functions.
- [Section 9.3](#) **Defining command procedures with SCIL**: The principles for defining command procedures using SCIL and an example.

9.1 General

9.1.1 Use

Command procedure objects contain a SCIL program of up to 2 000 000 program lines and a set of attributes which control the execution of the object and provide various information about the object. Command procedures can be started automatically or manually. They can be used for all kinds of automatic operations, for example, calculations, control operations, report printouts, automatic system and communication configuration.

Sometimes a command procedure object is used only as a storage for a SCIL program with no automatic activation mechanism.

9.1.2 Function

Execution of a command procedure object is carried out in the following steps:

1. If there is a compiled version of the SCIL program (attribute CP), it is executed, otherwise the source code (attribute IN) version is executed.
2. The status of the program execution (attribute OS) along with a time stamp (attributes RT and QT) is stored in the object.
3. The dynamic attributes OS, RT and QT are stored also on disk, unless the object is defined as "memory only" (attribute MO).

A command procedure is activated in the following three ways:

- By a SCIL program with commands #EXEC and #EXEC_AFTER (see [Section 4](#)), for example, #EXEC PROG:C.
- By a time channel ([Section 8](#)), giving an automatic time based execution.
- By an event channel ([Section 11](#)), giving an automatic event based execution triggered by a process or a system event.

It is possible to use all the three ways for activating the same command procedure.

When a command procedure object is used only as a storage for a SCIL program, the program is usually executed by the #DO command or DO function of SCIL language, for example, #DO PROG:CIN or #DO PROG:CCP.

As a rule, a command procedure object is executed in the same execution queue as its activating object (time channel, event channel or another command procedure). Command procedures activated by user interface programs are executed in the event channel queue. In addition, the command procedure object itself may specify parallel execution, see attributes PE and PQ.

9.1.3 Program

The SCIL program of a command procedure may be stored as the source code (text), as a compiled program (byte string), or as both. The command procedure object always executes the compiled version of the program, if it exists.

Because command procedures have no user interface, they are unable to handle user interface related SCIL commands (Visual SCIL commands, primitive graphics and picture commands).

9.1.4 Variables

Besides the variables defined by the command procedure program itself, the following variables can be used in the program:

- argument variables given as a variable list of #EXEC or #EXEC_AFTER command, when activated by SCIL (see [Section 3](#)).
- argument variables of the event channel, when activated by an event channel (see [Section 11](#)).
- variables of the time channel, when activated by a time channel. These are the variables created by the command procedures already executed by the time channel.

9.1.5 Storage

The combined maximum number of data and command procedure objects in an application is 2 000 000. Command procedure objects are stored in the report database files APL_REPORT.000 to APL_REPORT.399. When the application is WARM or HOT, the command procedures along with most of their attributes are stored in the global memory pool of SYS600 as well (see the System Objects manual).

However, the most memory consuming attributes (attributes IN and CP, which contain the SCIL program) are not permanently stored in the global memory pool. Instead, the most recently used programs are cached in the report cache along with the history data of the data objects. The memory space for the report cache is allocated from the global memory pool (see the System Objects manual, base system object attribute RC).

For optimization of the throughput, the dynamically changing attributes OS, QT and RT may be defined to be "memory only", see attribute MO. In this case, they are not stored in the disk file at all.

9.1.6 Object notation

Command procedures and their attributes are accessed from SCIL with the following object notation (see also [Section 4](#)):

name:{application}C{attribute}{(index)}

or

name:{application}C{attribute}{index}

where

'name'	is the name of the object
'application'	is the logical application number
'attribute'	is the attribute name
'index'	is an index or index range

Depending on the context, an object notation without an attribute refers to the entire object (for example in command #EXEC) or to the default attribute IN (for example in #SET command and as an operand of an expression).

9.2 Command procedure attributes

In this description, the command procedure attributes are grouped into the following sub-sections:

Section 9.2.1	Basic attributes	CM, FI, FX, IU, LN, ON, ZT
Section 9.2.2	Program	CP, CS, IN
Section 9.2.3	Time and validation stamps	OS, QT, RT, TS
Section 9.2.4	Execution control	EP, SE, TC, PE, PQ
Section 9.2.5	Storage attributes	HN, MO

9.2.1 Basic attributes

9.2.1.1 CM Comment

A freely chosen comment text.

Data type:	Text
Value:	Any Unicode text, up to 255 characters
Access:	No restrictions

9.2.1.2 FI Free Integer

This attribute is reserved for the SCIL application. It has no functionality in the base system.

Data type:	Integer
Default value:	0
Access:	No restrictions

9.2.1.3 FX Free Text

This attribute is reserved for the SCIL application. It has no functionality in the base system.

Data type:	Text
Value:	Any Unicode text, up to 255 characters
Default value:	""
Access:	No restrictions

9.2.1.4 IU In Use

The attribute states whether the object is in use or not. When the object is out of use ($IU = 0$), it is not executed. However, the attributes can be both read and written normally.

Data type:	Integer
Value:	0 Out of use
	1 In use
Default value:	0
Access:	No restrictions

9.2.1.5 LN Logical Name

The logical name of the command procedure.

Data type:	Text
Value:	Object name
Access:	Read-only, configurable

9.2.1.6 ON OPC Item Name

The alias name of the object to be used as an OPC item id.

Data type:	Text
Value:	Up to 255 Unicode characters. For the syntax, see below.
Default value:	""
Access:	No restrictions

The value of the attribute is a hierarchical name that consists of any number of fields separated by periods:

"field1.field2.fieldn"

Each field may contain any visible characters, except for colons and periods. The name is case-sensitive. Single embedded spaces are allowed in the field, whereas leading and trailing spaces are not accepted.

Examples of the valid ON attribute values:

- "South Tipperary.Kilkenny.Relay1.Breaker.Position"
- "Äänekosken ala-asema.Laukaan syöttö.Katkaisija.Tila"

The names are unique within an application. Consequently, the user cannot, for example, give the same ON attribute value to a process object and a command procedure object.

The hierarchy formed by the ON attribute values is shown by the name space browser of the OPC Server.

9.2.1.7 ZT Modification Time

The time when the object was created or modified. This attribute is set by the base system when the object is created, and it is updated each time the object is updated by the #MODIFY command (for example, by the command procedure object definition tool).

Data type:	Time
Access:	Read-only

9.2.2 Program

9.2.2.1 CP Compiled Program

The result of SCIL compilation of the source program (attribute IN). This is the program that is executed by the command procedure, unless it is empty.

When the IN attribute is modified alone (without CP attribute), the CP attribute is cleared.

Data type:	Byte string
Value:	Byte string value containing the byte code
Default value:	Empty byte string
Access:	Read only, configurable

9.2.2.2 CS Compilation State

Data type:	Integer
Value:	0 Not compiled (CP is empty)
	1 Compiled (CP is non-empty)
Access:	Read-only

Indicates whether the SCIL compiled version of the source program exists.

9.2.2.3 IN Instructions

The SCIL source program as a text vector. This program is executed by the command procedure if CP attribute is empty.

When the IN attribute is modified alone (without CP attribute), the CP attribute is cleared.

Data type:	Text vector
Value:	Up to 2 000 000 SCIL source code lines
Indexing:	Program line number, 1 ... 2 000 000.
Default value:	Empty vector
Access:	Read-only, configurable

9.2.3 Time and validation stamps

9.2.3.1 OS Object Status

The termination status of the latest execution of the command procedure.

Data type:	Integer
Value:	A SCIL status code.
Access:	Read, conditional write. It can be written with #SET.

9.2.3.2 QT Qualified Registration Time

The time of the latest execution of the command procedure. The attribute is set after the program execution is completed. The attribute gets its value from the system clock or from the variables QT, RT and RM, see "[TS Time Stamp](#)".

Data type:	List
Value:	Qualified time, attributes
CL	Time
MS	Integer, milliseconds
DS	Boolean, daylight saving
Access:	Read, conditional write. It can be written with #SET.

9.2.3.3 RT Registration Time

The time of the latest execution of the command procedure. The attribute is set after the program execution is completed. The attribute gets its value from the system clock or from the variable QT or RT, see "[TS Time Stamp](#)".

Data type:	Time
Value:	Execution time
Access:	Read, conditional write. It can be written with #SET.

9.2.3.4 TS Time Stamp

The TS attribute specifies whether the attributes RT and QT get their value from the system clock, or from the (argument) variables QT, RT and RM.

Data type:	Integer
Value:	0 RT and QT are read from the system clock
	1 RT and QT are read from variables QT, RT and RM
Default:	0
Access:	No restrictions

When TS = 1, the registration time of the object is set according to the following rules:

1. If argument variable QT is given and it contains a valid qualified time, it is used as the registration time.
2. Otherwise, if argument variable RT is given and it contains a time value, it is used as the registration time (possibly further specified by argument variable RM).
3. If neither a valid QT nor RT argument variable is given, the registration time is read from the system clock.

When the object is started by a process event channel or by a predefined event channel, argument variables QT, RT and RM normally contain the time of the triggering event. When TS attribute is set to 1, the RT and QT attributes of the command procedure object will match the time of the event.

9.2.4 Execution control

9.2.4.1 EP Execution Priority

The execution order of the object relative to other command procedures and data objects within the same time channel. Objects with the same EP value may be executed in any order.

Data type:	Integer
Value:	0 ... 255
	0 is the highest and 255 the lowest priority order
Default value:	255
Access:	Read-only, configurable

9.2.4.2 PE Parallel Execution

This attribute indicates whether the object is executed by a parallel queue or not, see [Section 6](#).

Data type:	Integer
Value:	0 Non-parallel execution
	1 Parallel execution
Default value:	0
Access:	No restrictions

PQ Parallel Queue

The number of the parallel queue that will execute the object, see [Section 8](#). If PE = 0, this attribute has no meaning.

Data type:	Integer
Value:	0 ... APL:BPQ
	Value 0 means that any parallel queue will do.
Default value:	0
Access:	No restrictions

9.2.4.4 SE Start-up Execution

This attribute indicates whether the command procedure is executed during application start-up, when reporting is chasing the real time. If the value of this attribute is 0 (no start-up execution) and a scheduled run has been missed during the application shutdown, the status attribute (OS) is set to NOT_SAMPLED_STATUS (10) and the time stamp attributes (RT and QT) tells the time of last missed execution.

Data type:	Integer
Value:	0 No
	1 Yes
Default value:	0
Access:	No restrictions

9.2.4.5 TC Time Channel

The name of the time channel that runs the command procedure.

Data type:	Text
Value:	Object name
Access:	Read-only, configurable

9.2.5 Storage attributes

9.2.5.1 HN History File Number

Command procedures are stored in files named APL_REPORT.nnn, where "nnn" is a three-digit sequence number. This file number is specified by the HN attribute. The file numbers can be freely chosen from the range 0 ... 399.

Data type:	Integer
Value:	0 ... 399
Default value:	0
Access:	Read-only, configurable

Note: Division of the report database into several files was, in the first place, implemented due to the 32 MB size limit of MicroSCADA files. Now that no such restriction exists anymore, using HN values other than 0 is more or less obsolete.

9.2.5.2 MO Memory Only

This attribute determines whether the dynamically changing attributes OS and RT (QT) are stored on disk. If old values (stored before last start-up) of these attributes are not interesting to the application, some performance gain may be achieved by setting this attribute to 1.

Data type:	Integer
Value:	0 OS and RT (QT) are stored on disk
	1 OS and RT (QT) are stored only in RAM
Default value:	0
Access:	Read-only, configurable

9.3 Defining command procedures with SCIL

The minimum number of attributes to be defined when creating a new command procedure is the logical name, the LN attribute.

Other attributes get their default values mentioned in the attribute descriptions above.

The following SCIL program example creates a command procedure called LIST for the printout of a process object:

```
#CREATE LIST:C = LIST(IN = VECTOR("#LIST 2 'LN':P"), IU = 1)
```

Section 10 Time channels

This section describes the time channels, the time channel attributes and the definition of time channels. It contains the following sections:

- [Section 10.1](#) **General:** The use and function of time channels, the object notation and the storage of time channels.
- [Section 10.2](#) **Time channel attributes:** The time channel attributes listed and described. The attributes are grouped according to their function.
- [Section 10.3](#) **Defining time channels with SCIL**

10.1 General

10.1.1 Use

Time channels provide schedules for automatic time activated start-up of operations in the report database: logging of data objects and execution of command procedures. A time channel can start one or more objects. If a time channel starts several objects, they are started in priority order, see [Figure 12](#). Each data and command procedure object can be connected to only one time channel at a time.

A time channel is activated at chosen times, either at an absolute moment or cyclically at fixed intervals. Discontinuous time activation is handled by means of conditions.

Time channels are used for cyclic program execution or data registration: time dependent reports, trends, regular checks, time control, etc.

10.1.2 Function

A time channel has two functions: **execution** and **initialization**, see [Figure 12](#). Execution of a time channel means that the data and command procedure objects connected to the time channel are executed. The objects are executed in the order of their priority (the EP attribute). Initialization implies that the history registrations of the data objects attached to the time channel are cleared (see [Section 9](#) for details). For command procedures, initialization has no meaning.

Both initialization and execution can take place cyclically with fixed cycle lengths. The cycle may be synchronized to the calendar clock or not. If not, adjusting the system clock does not affect the effective length of the cycle (see attribute CP).

Cyclic initialization and execution is synchronized at chosen **synchronization** times. At synchronization the cyclic initialization/execution is restarted, regardless of the phase of the cycle in progress. Hence, initialization/execution always takes place at the synchronization times. If no cycle is given, initialization/execution is done only at synchronization times. Synchronization can occur once at a selected time or periodically once a year, once a month, once a week, once a day or once an hour or it may be tied to a daylight saving time switch, see [Figure 12](#).

Conditional initialization and execution is obtained by means of **conditions**. A condition is a boolean SCIL expression that is evaluated before the time channel is run. If the evaluation results to TRUE, the time channel is run, otherwise it is skipped.

When run times of time channels coincide, the following rules are applied, in the given order, to resolve the running order:

1. All executions are always done before initializations.
2. Runs synchronized to daylight saving time switch are done before others.
3. Runs with shorter cycles are done first.
4. Runs with shorter synchronization periods are done first.

If the rules do not resolve the order, the runs take place in unspecified order.

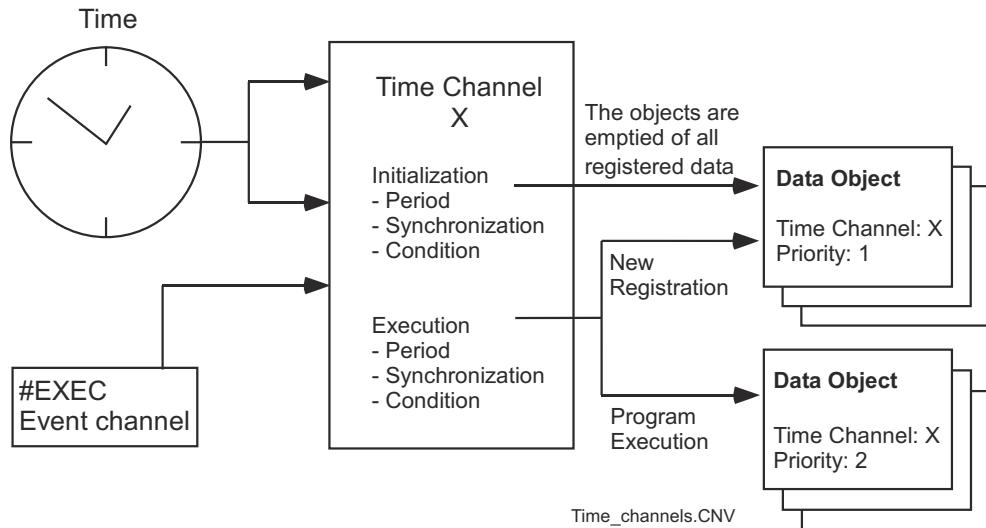


Figure 12: The function of time channels. Data objects and command procedures are started in the order of their priority (the EP attribute).

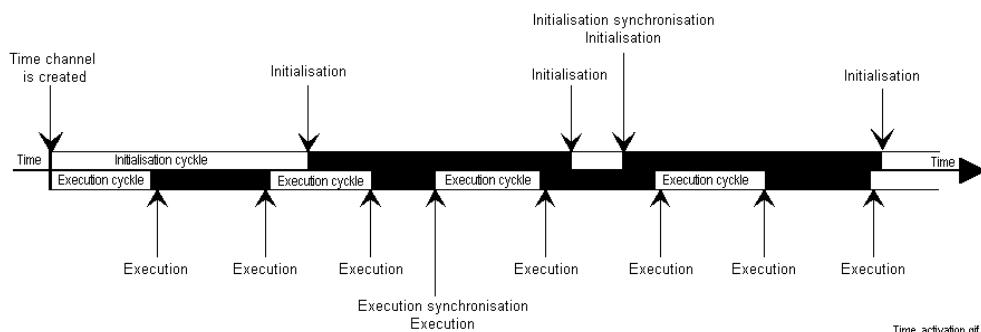


Figure 13: The activation of a time channel with periodic execution and initialization.

Time channels can also be executed by event channels ([Section 11](#)) and by the #EXEC and #EXEC_AFTER command ([Section 5.3](#)). In these cases, execution of the time channel takes place immediately and independent of the condition. The execution does not affect the scheduled behavior of the time channel. Initialization of the time channel cannot be done this way.

When an application is restarted after a shutdown, time channels are scheduled according to their start-up execution policy, either once for each missed run, once or only according to their next run time (see attribute SP).

As a rule, time channels are run in the time channel queue of the report database, as the name of the queue implies. However, if parallel execution is specified (attribute PE is set to 1), the time channel queue only takes care of the scheduling, the execution is done in one of the parallel queues. Also, the executions started by an event channel or by SCIL are done in the event channel queue.

10.1.3 Storage

The maximum number of time channels in an application is 65 535. Time channel objects are stored in the report database file APL_REPORT.000. When the application is WARM or HOT, the time channels along with all their attributes are stored in the global memory pool of MicroSACADA as well (see the System Objects manual). A time channel may be defined as "memory only" (see attribute MO). In this case, the dynamically changing attributes (time tagging attributes) are not written into the disk file.

10.1.4 Object notation

Time channel attributes are accessed in SCIL with the following object notation (see also [Section 4](#)):

`name:[application]T[attribute]{(index)}`

or

`name:[application]T[attribute]{index}`

where

'name'	is the name of the time channel
'application'	is the logical application number
'attribute'	is the attribute name
'index'	is an index or an index range

Time channels have no default attribute. Consequently, a time channel object notation without an attribute name always refers to the entire object.

Indexing is used to distinguish between initialization and execution.

10.2 Time channel attributes

The time channel attributes are grouped into the following subsections:

Section 10.2.1	Basic attributes	CM, IU, LN, MO, ZT
Section 10.2.2	Scheduling	CD, CP, CY, DP, SP, SU, SY
Section 10.2.3	Parallel execution	PE, PQ, SX
Section 10.2.4	Time tagging	QB, QE, QS, QT, RB, RE, RS, RT

10.2.1 Basic attributes

10.2.1.1 CM Comment

A freely chosen text.

Data type:	Text
Value:	Any Unicode text, up to 255 characters
Access:	No restrictions

10.2.1.2 IU In Use

This attribute indicates whether the time channel is in use or not. When the time channel is not in use ($IU = 0$), it is never executed. However, its attributes can be both read and written normally. When a time channel is taken into use (IU is set from 0 to 1), it is scheduled according to the values of its scheduling attributes.

Data type:	Integer	
Value:	0	Out of use
	1	In use
Default value:	0	
Access:	No restrictions	

10.2.1.3 LN Logical Name

The name of the time channel.

Data type:	Text	
Value:	Object name	
Access:	Read-only, configurable	

10.2.1.4 MO Memory Only

Specifies whether the dynamically changing attributes (time tagging attributes, see [Section 10.2.4](#)) are written into the disk database file. Disabling disk writes may be useful when running SYS600 without a hard disk.

Data type:	Integer	
Value:	0	All attributes are stored on disk
	1	Time tagging attributes are not stored on disk
Access:	No restrictions	

At application start-up, the system does not know when a memory-only time channel has been executed last. Therefore, the start-up execution policy value "YES" (see attribute SP) is not allowed while $MO = 1$.

10.2.1.5 ZT Modification Time

The time when the object was created or modified. This attribute is set by the base system when the object is created, and it is updated each time the object is updated by the #MODIFY command (for example, by an object definition tool).

Data type:	Time
Access:	Read-only

10.2.2 Scheduling

10.2.2.1 CD Condition

Condition for running the time channel.

Data type:	Vector of 2 text elements
Value:	Boolean SCIL expression
Indexing:	1 Initialization 2 Execution
Default value:	Empty string (no condition)
Access:	No restrictions

10.2.2.2 CP Cycle Policy

Specifies whether the cycle (attribute CY) is synchronized to the calendar clock or not.

If the cycle policy is set to "CALENDAR", the execution and initialization run time is synchronized to the system clock. For example, if the cycle is 15 minutes, the scheduling takes place at 00:00, 00:15, 00:30 etc. (unless synchronized to some other minute using SY attribute). If the system clock is adjusted, the effective cycle length may differ from the specified 15 minutes. The behavior at daylight saving time switches is specified by the DP attribute.

If the cycle policy is set to "EVEN", the run times are not synchronized to the system clock. If the cycle is 15 minutes, the scheduling may take place at 00:03:37, 00:18:37, 00:33:37, etc. Even if the system clock is adjusted, the effective cycle length is 15 minutes. Daylight saving time switches do not affect the cycle length.

Data type:	Text keyword	
Value:	"CALENDAR"	Synchronized to calendar clock
	"EVEN"	Not synchronized
Default value:	"CALENDAR"	
Access:	No restrictions	

10.2.2.3 CY Cycle

Time interval between cyclic runs. The cycling starts at synchronization times.

Data type:	Vector of 2 integers	
Value:	Cycle length, >= 0	
Unit:	Seconds	
Indexing:	1 Initialization 2 Execution	
Access:	No restrictions	

10.2.2.4 DP Daylight Switch Policy

Specifies the behavior of the time channel, when the local time of the system switches from standard time to daylight saving time or vice versa.

If DP is set to "CALENDAR", the scheduling is synchronized to the local time, if "EVEN", it is synchronized to the UTC time.

If the system runs in UTC time, this attribute has no effect, because there is no daylight saving time. Also, when CP is set to "EVEN", this attribute has no meaning.

Data type:	Text keyword	
Value:	"CALENDAR"	Scheduled according to local time
	"EVEN"	Scheduled according to UTC time
	"APPLICATION_DEFAULT"	see below
Default value:	"EVEN"	
Access:	No restrictions	

Attribute value "APPLICATION_DEFAULT" is designed for compatibility. Time channels created by MicroSCADA revisions earlier than 8.4.4 have this value for their DP attribute. These time channels are normally scheduled according to EVEN policy. However, if the revision compatibility switch (APL:BRC) "DEFAULT_DAYLIGHT_POLICY_IS_CALENDAR" is set, the time channels created by MicroSCADA revisions earlier than 8.4.4 keep their behaviour, that is, they are scheduled according to CALENDAR policy. See manual the System Objects manual.

Example:

Suppose that there are two time channels, both having a cycle of 30 minutes and CP set to "CALENDAR". The first have its DP set to "CALENDAR" and the second one to "EVEN". Suppose also that the user lives in time zone 2 (2 hours ahead of UTC time) and the switch from standard time (STT) to daylight saving time (DST) occurs at 03.00 in the spring by moving the clock one hour forward, and the switch back to standard time occurs at 04.00 in autumn by moving the clock one hour backward. Now, the two time channels are scheduled as follows:

UTC	Local	TC1	TC2	
00:00	02:00 STT	X	X	Spring
00:30	02:30 STT	X	X	
01:00	03:00	X		From STT to DST
01:00	03:30	X		
01:00	04:00 DST	X	X	
01:30	03:00 DST	X	X	
...	...			
00:00	03:00 DST	X	X	Autumn
00:30	03:30 DST	X	X	
01:00	03:00 STT		X	From DST to STT
01:30	03:30 STT		X	
02:00	04:00 STT	X	X	

10.2.2.5 SP Start-up Execution Policy

Specifies how the time channel is scheduled at application start-up.

When an application is started, time channels are scheduled for runs that have been missed during down-time of the application. This stage is called "real time chase". SP attribute specifies whether the time channel is scheduled during the real time chase.

Data type:	Text keyword	
Value:	"NO"	Not scheduled during real time chase
	"YES"	Scheduled during real time chase

Table continues on next page

	"ONCE"	Scheduled once, immediately after the real time chase is completed
Default value:	"YES"	
Access:	No restrictions	

Note: For memory-only (MO = 1) time channels, SP attribute value "YES" is not allowed, because the system does not know when the last run took place.

10.2.2.6 SU Synchronization Unit

Defines how often periodic synchronization takes place. The exact moment of first synchronization is defined by the SY attribute.

Data type:	Vector of 2 integers
Value:	Synchronization unit coded as 0 ... 7:
	0 No synchronization or once at the time determined by the SY attribute
	1 Once a year at the time determined by the SY attribute
	2 Once a month at the time determined by SY
	3 Last day of month at the time determined by the SY attribute
	4 Once a week on the day of week and time fixed by the SY attribute
	5 Once a day at the hour and minute determined by the SY attribute
	6 Once an hour at the minute determined by the SY attribute
	7 CY seconds before next daylight saving time switch
Indexing:	1 Initialization
	2 Execution
Default value:	0
Access:	No restrictions
Example:	The synchronization time has been set to 1991-02-14 13:40:38. If SU = 2 (synchronization once a month), synchronization occurs at 13:40:38 o'clock on the 14th of every month.

10.2.2.7 SY Synchronization Time

The time of first synchronization. At the synchronization time, periodic initialization/execution is started or restarted. See the SU attribute.

Data type:	Vector of 2 time or list elements (qualified time)
Value:	Absolute time
Indexing:	1 Initialization
	2 Execution
Default value:	1978-01-01 00:00 (zero time)
Access:	No restrictions



If the given time is ambiguous, that is, it falls into the one-hour period that is lived twice during the transition from the daylight saving time to the standard time, the first occurrence (daylight saving time) is assumed.

Example:

Time channel X is set to be executed after an hour. After that it is scheduled according to its other scheduling attributes.

```
#SET X:TSY2 = CLOCK + 3600
```

10.2.3 Parallel execution

10.2.3.1 PE Parallel Execution

This attribute indicates whether the object is executed in a parallel queue or not, see [Section 6](#).

Data type:	Integer	
Value:	0	Non-parallel execution
	1	Parallel execution. The object is executed in the queue specified by the PQ attribute.
Default value:	0	
Access:	No restrictions	

10.2.3.2 PQ Parallel Queue

The number of the parallel queue, meaningful only if PE = 1. See [Section 6](#).

Data type:	Integer	
Value:	0 ... APL:BPQ	
Value 0 means that any parallel queue will do.		
Default:	0	
Access:	No restrictions	

10.2.3.3 SX Synchronized Execution

Specifies whether the completion of parallel objects of the time channel is waited for before the next non-parallel object is started, see [Figure 14](#).

Data type:	Integer	
Value:	0	No
	1	Yes
Default value:	0	
Access:	No restrictions	

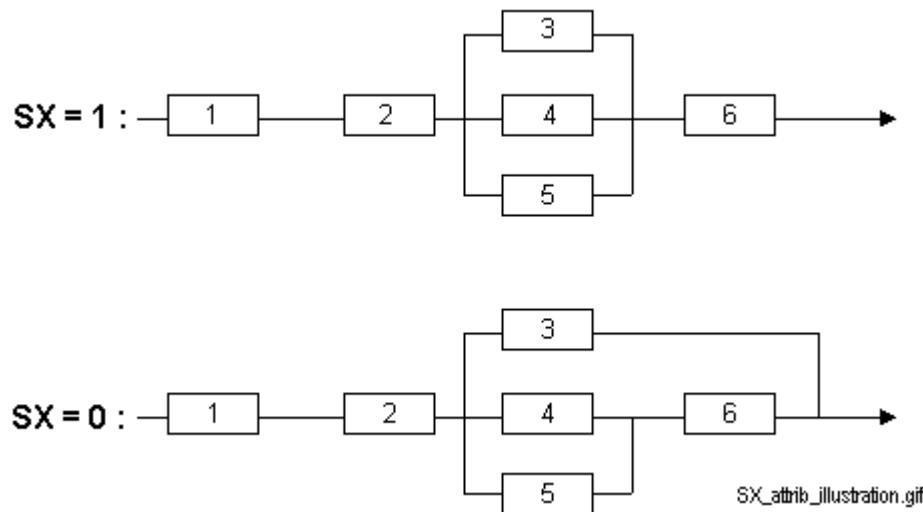


Figure 14: An illustration of the SX attribute. The numbered boxes represent data objects and command procedures that are run by the time channel. Objects 3, 4 and 5 are parallel, the others are nonparallel. SX = 1 should be chosen if object number 6 is dependent on the results from objects 3, 4 and 5.

10.2.4 Time tagging

Time tagging attributes contain the timing information of latest scheduled run (execution and initialization) of the time channel. Attributes RT, RS, QT and QS indicate scheduled times, that is, the times when the time channel was supposed to be run. Attributes RB, RE, QB and QE tell the realized run time. The latter ones are updated only if the condition (attribute CD) of the time channel is fulfilled, that is, the data and command procedure objects of the channel are executed (or initialized).

Attributes RT, RS, RB and RB give the run times with one second resolution as time data type. Attributes QT, QS, QB and QE supply one millisecond resolution and daylight saving time information as well.

The time tagging attributes are updated only by scheduled runs. They do not change when a time channel is executed via an event channel or by SCIL command #EXEC or #EXEC_AFTER.

10.2.4.1 QB Qualified Begin Time

QB attribute registers the realized start time of the latest scheduled run of the time channel. It is set before the first data or command procedure object of the time channel is executed (or initialized). If the condition of the time channel is not fulfilled, the attribute value does not change.

Data type:	Vector of 2 list elements
Value:	Qualified time, attributes
	CL Time
	MS Integer, milliseconds
	DS Boolean, daylight saving
Indexing:	1 Initialization
	2 Execution
Access:	Read only

10.2.4.2 QE Qualified End Time

QE attribute registers the realized completion time of the latest scheduled run of the time channel. It is set after the last data or command procedure object of the time channel has been executed (or initialized).

Data type:	Vector of 2 list elements
Value:	Qualified time, attributes
CL	Time
MS	Integer, milliseconds
DS	Boolean, daylight saving
Indexing:	1 Initialization 2 Execution
Access:	Read only

10.2.4.3 QS Qualified Synchronization Time

The latest scheduled synchronization time. This attribute is updated even if the condition of the time channel is not fulfilled. Because scheduling is done once a second, the millisecond part of the time is always zero.

Data type:	Vector of 2 list elements
Value:	Qualified time, attributes
CL	Time
MS	Integer, milliseconds
DS	Boolean, daylight saving
Indexing:	1 Initialization 2 Execution
Access:	Read only

10.2.4.4 QT Qualified Registration Time

The latest scheduled run time. This attribute is updated even if the condition of the time channel is not fulfilled. Because scheduling is done once a second, the millisecond part of the time is always zero.

Data type:	Vector of 2 list elements
Value:	Qualified time, attributes
CL	Time
MS	Integer, milliseconds
DS	Boolean, daylight saving
Indexing:	1 Initialization 2 Execution
Access:	Read only

10.2.4.5 RB Registered Begin Time

RB attribute registers the realized start time of the latest scheduled run of the time channel. It is set before the first data or command procedure object of the time channel is executed (or

initialized). If the condition of the time channel is not fulfilled, the attribute value does not change.

Data type:	Vector of 2 time elements
Indexing:	1 Initialization
	2 Execution
Access:	Read only

10.2.4.6 RE Registered End Time

RE attribute registers the realized completion time of the latest scheduled run of the time channel. It is set after the last data or command procedure object of the time channel has been executed (or initialized).

Data type:	Vector of 2 time elements
Indexing:	1 Initialization
	2 Execution
Access:	Read only

10.2.4.7 RS Registered Synchronization Time

The latest scheduled synchronization time. This attribute is updated even if the condition of the time channel is not fulfilled.

Data type:	Vector of 2 time elements
Indexing:	1 Initialization
	2 Execution
Access:	Read only

10.2.4.8 RT Registration Time

The latest scheduled run time. This attribute is updated even if the condition of the time channel is not fulfilled.

Data type:	Vector of 2 time elements
Indexing:	1 Initialization
	2 Execution
Access:	Read only

10.3 Defining time channels with SCIL

When defining a new time channel, the logical name (LN) is obligatory. The rest of the attributes get the default values mentioned above.

Example:

```
V = FETCH(2, "T", "TC_1")
V.CD = ("", "DOW == 7")
#CREATE TC_2:T = V
```

A new time channel is created by copying an existing one from application 2 and adding a condition for execution (executed only on Sundays).

Section 11 Event channels

This section describes event channels and their attributes, as well as how to define the event channels. It is divided in the following three sections:

- | | |
|------------------------------|--|
| Section 11.1 | General : The use of event channels and their function, the variables transferred to the event channels. |
| Section 11.2 | Event channel attributes : Event channel attributes in alphabetical order. |
| Section 11.3 | Predefined event channels : Predefined event channels. The event channels with predefined names and functions. |

11.1 General

11.1.1 Use

Event channels are facilities for automatic event activated start-up of operations in the report database. An event channel can start execution of data objects, command procedures and time channels. Event channels are activated by process events (changes in the process object values), by some application and system events and by SCIL. In other words, they create a channel between external events and the report database.

The event channels are used for, for example:

- Storing process events in the report database.
- Event activated program execution, for example process control, calculation, printout, etc.
- Forwarding process data to other objects.
- Automatic dial-up of stations or remote workstations.
- Event-activated system configuration.

11.1.2 Function

An event channel is essentially a list of other type of report database objects (command procedures, data objects and time channels). One of these objects is the primary object and the others (up to 10) are called secondary objects. The primary object is mandatory, secondary objects are optional.

Activation (execution) of an event channel means that the primary object is first queued for execution, followed by the secondary objects (if any) in the order they are listed in the event channel object. Note that the objects are not executed at once, they are only queued to one of the report database queues to be executed as soon as possible.

As a rule, the objects are queued to the event channel queue of the report database, as the name of the queue implies. The objects may, however, specify parallel execution instead (attribute PE set to 1). If an event channel is executed by SCIL in a command procedure object, the objects are queued to the queue where the command procedure is run.

Event channels are activated in three ways:

1. By a process object defined with an event channel (the AN attribute). Certain changes of process object attributes activate the event channel (see below).
2. By some predefined system events (see [Section 11.3](#)).
3. Using the SCIL command #EXEC or #EXEC_AFTER.

11.1.3 Process event channels

An event channel is activated automatically by a change of the state of a process object, if the following criteria is met:

1. an event channel (attribute AN) is defined for the object
2. activation is enabled (attribute AE)
3. the activation criterion (attribute AA) is fulfilled

Examples of activation criteria are a change of the value (attribute OV) of the object, change of the alarm state (on/off) and a change of the alarm zone of an analog input object. For details, see [Section 5.2.8](#).

A set of argument variables are passed to the activated event channel. These attributes are called snapshot variables, they contain a snapshot of the state of the object at the moment of the event. For a complete list of these variables, see [Section 5.2.8](#).

Event Channel Activation

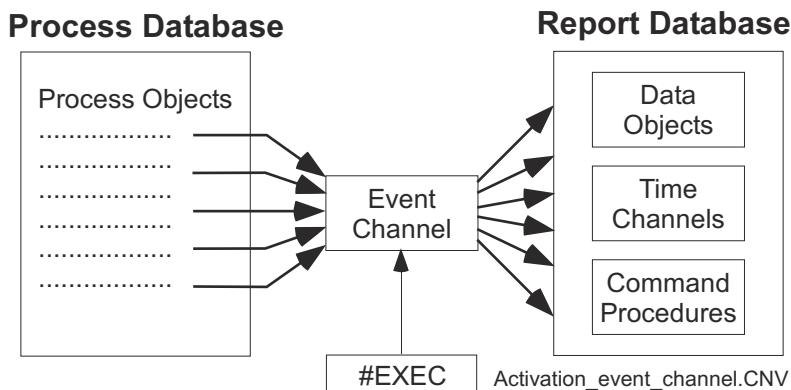


Figure 15: The activation of an event channel

11.1.4 Predefined event channels

Each application can contain a number of event channels with predefined names and activating events. Examples of these are event channels activated at application start-up, by alarm and by various system events. These event channels are described in [Section 11.3](#).

11.1.5 Variables

When an event channel is activated by SCIL, any number of named arguments may be passed to it as a variable list.

Example:

```
#EXEC CHANNEL:A (@NAME = "ABC", @OV = 1)
```

The command procedure and data object(s) started by the channel may use the argument variables as any other variables, and they are preset when the object starts its execution. Each

object started by the event channel has the same initial values for the argument variables. If, for example, event channel CHANNEL in the example starts command procedures C1 and C2, and C1 modifies variable OV, C2 still starts with OV value 1.

When an event channel is activated automatically by the process database, the argument variables, called snapshot variables, are predefined and, as a rule, have the name of a process object attribute. They reflect the state of the process object at the very moment of the process event. The snapshot variables are described in [Section 5.2.8](#).

Argument variables of predefined event channels started by various system events are described in [Section 11.3](#).

11.1.6 Storage

The maximum number of event channels in an application is 65 535. Event channel objects are stored in the report database file API_REPORT.000. When the application is WARM or HOT, the event channels, along with all their attributes, are stored in the global memory pool of SYS600 as well (see the System Objects manual).

11.1.7 Object notation

Event channel attributes are accessed in SCIL with the following object notation (see also [Section 4](#)):

name:[application]A{attribute}{(index)}

or

name:[application]A{attribute}{index}

where

'name' is the name of the event channel

'application' is the logical application number

'attribute' is the attribute name

'index' is an index or an index range

Event channels have no default attribute. Consequently, an event channel object notation without an attribute name always refers to the entire object.

11.2 Event channel attributes

11.2.1 CM Comment

A freely chosen comment text.

Data type: Text

Value: Any Unicode text, up to 255 characters

Access: No restrictions

11.2.2 LN Logical Name

The name of the event channel.

Data type: Text
Value: Maximum length 63 characters.
Access: Read-only, configurable

11.2.3 ON Object Name

The name of the primary object activated by the event channel.

Data type: Text
Value: Object name. Maximum length 63 characters.
Access: No restrictions

11.2.4 OT Object Type

The type of the primary object activated by the event channel.

Data type: Text
Value:
 "D" Data object
 "C" Command procedure object
 "T" Time channel object
Access: No restrictions

11.2.5 SN Secondary Object Names

The names of the secondary objects (up to 10) that are activated by the event channel. The corresponding object types must be given in the ST attribute.

Data type: Text vector
Value: Up to 10 object names
Indexing: Secondary object number. The same indexing is used by the ST attribute.
Access: Read only, configurable

11.2.6 ST Secondary Object Types

The types of secondary objects activated by the event channel.

Data type: Text vector
Value: A vector of the same length as SN containing the types of objects:
 "D" Data object
 "C" Command procedure object
 "T" Time channel object
Indexing: Secondary object number. The same indexing as in the SN attribute.
Access: Read only, configurable

11.2.7 ZT Modification Time

The time when the object was created or modified. This attribute is set by the base system when the object is created, and it is updated each time the object is updated by the #MODIFY command (for example, by an object definition tool).

Data type:	Time
Access:	Read only

11.3 Predefined event channels

Each application can contain a number of event channels that are activated by certain process, application or system events. These event channels have predefined names and predefined argument variables. If the application wants to respond to these events, it should create the corresponding event channel objects and the objects (typically command procedures) connected to the event channels.

There are two kinds of predefined event channels: application specific and system wide event channels. Each application in the system has its own (independent) application specific event channels for its events. In a multi-application system, the applications should coordinate about responding to system events. Typically, only one application responds to the predefined system events.

The predefined application and system event channels and their argument variables are described below.

11.3.1 Predefined application event channels

11.3.1.1 Application start-up and shutdown event channels

The following three event channels are activated at different stages of application start-up:

1. APL_INIT_1, activated when the application databases have been set up but execution of time channels has not yet been started.
2. APL_INIT_2, activated when the time channels have reached the real time, that is, when the reporting system runs in real time.
3. APL_INIT_H, activated when the application that becomes hot after a hot-stand-by take-over is started. In this case, event channels APL_INIT_1 and APL_INIT_2 are not activated.

Unlike the other event channels, these three event channels are run in the time channel queue of the report database.

Shutdown event channel APL_CLOSE is activated just before the state of the application changes from HOT to WARM. After the objects activated by the event channel (both the primary and the secondary objects, normally command procedures) have all completed, the state of the application is set to WARM and the application shutdown is completed.

These event channels have no argument variables.



The command procedures activated by APL_CLOSE shall not activate any further command procedures, because the application is no longer HOT.

11.3.1.2 Alarm event channel

Event channel APL_ALARM is activated for each alarm on or alarm off event in the process database.

The following argument variables are passed from the process object to the event channel:

LN	Logical name of the object
IX	Index of the object
AC	Alarm class of the object
AL	Alarm
AR	Alarm Receipt
AS	Alarm State
AT	Alarm Time
AM	Alarm Milliseconds
AQ	Alarm Qualified Time
US	Name of the user causing the event

11.3.1.3 Alarm acknowledgement event channel

Event channel APL_ALARM_ACK is activated every time an alarm is acknowledged (the AR attribute of a process object is set to 1).

The following argument variables are passed from the process object to the event channel:

LN	Logical name of the object
IX	Index of the object
AC	Alarm class of the object
AL	Alarm
AR	Alarm Receipt
AS	Alarm State
AT	Alarm Time
AM	Alarm Milliseconds
AQ	Alarm Qualified Time
US	Name of the user acknowledging the alarm

11.3.1.4 Blocking event channel

Event channel APL_BLOCKING is activated when a blocking attribute of a process object is set or cleared.

The following argument variables are passed from the process object to the event channel:

LN	Logical name of the object
IX	Index of the object
CA	Changed attribute (AB, HB, PB, XB or UB)
BV	New value of the attribute (0 or 1)
US	Name of the user changing the attribute

11.3.1.5 Monitor event channel

Event channel MON_EVENT is activated whenever a SYS600 user logs in or out or a communication break between the base system and a remote monitor is detected.

The following argument variables are passed to the event channel:

VIDEO_NR	Logical monitor number
MO	Physical monitor number
EVENT	The event that caused the activation:
1	Login
2	Logout
3	Communication break

11.3.1.6 Undefined process object

Event channel UNDEF_PROC is activated when an undefined (missing) process object is detected. A process object is considered missing if the process station sends an update message using an object address that is not found in the process database.

The following argument variables are passed to the event channel:

UN	Unit (logical station number)
OA	Object address

11.3.1.7 Undefined OPC event source

Event channel UNDEF_OPC_EVENT is activated when an undefined (missing) OE (OPC Event) type process object is detected. A process object is considered missing if the OPC A&E server sends an event notification with a source/category/event type/condition combination that has no matching OE process object counterpart in the process database.

The following argument variables are passed to the event channel:

UN	Unit (logical station number)
IN	Event source name
CI	Category ID
ET	Event type ("SIMPLE", "TRACKING" or "CONDITION")
CN	Condition name

11.3.1.8 Invalid OPC item

Event channel INVALID_OPC_ITEM is activated when the subscription or an update of a process object fails for reason or another (process objects of OPC type stations only).

The following argument variables are passed to the event channel:

UN	Unit (logical station number)
LN	Name of the process object
IX	Index of the process object
IN	Item name

Table continues on next page

IG	Group name
ERROR	Keyword describing the reason of the failure:
"BAD_SYNTAX"	The syntax of the item name is invalid.
"NON_EXISTENT"	The item does not exist in the server's name space.
"NOT_READABLE"	The item is not readable.
"NOT_WRITEABLE"	The item is not writable.
"NO_IG_ALLOWED"	IG attribute is not allowed in output objects.
"GROUP_NOT_FOUND"	The group specified by IG has not been created.
"DATATYPE_ERROR"	Data has been received that cannot be stored in the process object.
"FAILURE"	Unspecified error during OPC communication.

11.3.1.9 Mirroring configuration error

Event channel MIRRORING_CONFIGURATION is activated when the base system detects an error in the mirroring configuration. This event channel is implemented to ease finding unit (station) configuration errors in a system architecture based on the mirroring concept.

The following argument variables are passed to the event channel:

UNIT	Unit (logical station number)
TIME	The time stamp of the event (qualified time)
ERROR	Keyword describing the error:
"DOUBLE_SOURCE"	The unit receives input from two sources: from a host application and directly from the process.
"HSB_CONF_MISMATCH"	The mirroring attributes of the STA object do not match in the HSB system. This error is detected after a HSB switch-over if the STA object does not match with the data received through shadowing.
"DOUBLE_HOST"	Two units in different applications are configured to act as hosts of the same image unit. This event is sent to both applications.

The following argument variables are passed only when ERROR = "DOUBLE_HOST":

IMAGE_APL	The application number of the (external) image application.
IMAGE_UNIT	The unit number of the image unit.
HOST_APL	The application number of the (local) application causing the conflict.
HOST_UNIT	The unit number in the conflicting application.

11.3.1.10 Host address missing

Event channel HOST_ADDRESS_MISSING is activated in an image application, when the application subscribes to an object and the host reports that the object address does not exist in its database. This event channel is implemented to ease finding signal configuration errors in a system architecture based on the mirroring concept.

The following argument variables are passed to the event channel:

UN	Unit (logical station number)
OA	Object address (OA) (non-OPC units only)
OB	Object bit address (non-OPC units only)
IN	Item name (OPC and OAE units only). OAE (OPC Alarms and Events) units: For conditions events, the IN (Item Name) attribute of the object and the condition name combined as "ItemName\ConditionName". For other events, the IN attribute and the category id combined as "Item_Name\5". OPC (OPC Data Access) units: The IN attribute of the object postfixed by "IN" or "OUT", according to the polarity of the object, for example, "ItemName\OUT".

11.3.1.11 AEP_EVENT

Event channel AEP_EVENT is activated whenever the state of an Application Extension Program (AEP) changes. This event channel is implemented to supervise the AEP's of the application, for example to generate an alarm when an AEP becomes non-responsive.

The following argument variables are passed to the event channel:

AEPN	The (SCIL) number of the AEP
EVENT	Keyword describing the event:
	"START" The AEP has been started.
	"STOP" The AEP has stopped.
	"CRASHED" The AEP has terminated abnormally.
	"NO_RESPONSE" The AEP no longer responds.
	"RECOVERED" The AEP has recovered from the NO_RESPONSE state.
TIME	The time stamp of the event (qualified time)
START_COMMAND	The AEP_START argument used to start the AEP
ARGUMENT	The SCIL argument passed to the AEP
PROCESS_NAME	The name of the Windows process running as an AEP.
PROCESS_ID	The process id (PID) of the Windows process.

11.3.1.12 HISTORIAN_EVENT

Event channel HISTORIAN_EVENT is activated whenever the state of a Historian database connection changes or errors in configuration are encountered.

The following argument variables are passed to the event channel:

DATABASE	The name of the DATABASE logging profile object that defines the connection to the Historian database.
EVENT	Keyword describing the event:
	"CONNECTED" The database has been successfully connected.
	"RECONNECTED" The database has been successfully reconnected after a communication break. No data has been lost.
	"LOST" The connection to the database has been lost due to a communication break.
	"DISCONNECTED" The connection to the database has been closed. If disconnected by the communication process (CPMW), the reason is described by variable MESSAGE.

Table continues on next page

"STOPPED"	The communication process (CPMW) has stopped due to an error described by variable MESSAGE.
"CONNECT_FAILURE"	A connection to the database could not be established due to an error described by variable MESSAGE.
"TAG_NOT_FOUND"	A previously configured tag no longer exists in the database. The name of the tag is given as variable TAG. A new tag is created.
"MISSING_TEMPLATES"	The history templates given as text vector variable TEMPLATES do not exist in the Historian database.
TIME	The time stamp of the event (qualified time)

11.3.1.13 UAL_EVENT

Event channel UAL_EVENT is activated whenever an event is written to the User Activity Log.

The following argument variables are passed to the event channel:

EVENT_ID	UAL defined integer that identifies the type of the event
EVENT_TEXT	Localized description of the event id
TIME	Local time of the event as qualified time (data type list)
SOE	Sequence number of the event
USER	User name
SEVERITY	Syslog severity of the event (0 .. 7)
SOURCE	Source of the event (text)
SENDER_IP	IP address of the sender of the event
EXTRA_INFO	Event id specific extra info of the event

11.3.1.14 AOR_EVENT

Event channel AOR_EVENT is activated when a significant state change happens in an Area of Responsibility (AoR). When Exclusive Access Rights (EAR) are enabled, every change of responsible operator is reported.

The following argument variables are passed to the event channel:

AOR	Name of the AoR
USER	Name of the user that caused the event. Typically this is an operator that logged in or out, or a master operator that forced the responsibility to another operator. It may also be an administrator that caused the event by changing the AoR configuration.
EVENT	<ul style="list-style-type: none">"AOR_UNCONTROLLED" AoR is left uncontrolled (no AoR operator logged-in)"AOR_CONTROLLED" AoR is controlled (an AoR operator of an uncontrolled AoR logged in)"EAR_ASSIGNED" Exclusive Access Rights assigned to user TO"EAR_UNASSIGNED" Exclusive Access Rights unassigned (previous EAR owner user FROM logged out and no AoR operator logged-in)"EAR_REASSIGNED" Exclusive Access Rights reassigned from user FROM to user TO
FROM	Name of the EAR owner before this event
TO	Name of the EAR owner after this event

11.3.1.15 Generic application event channel

APL_EVENT is a generic predefined event channel used to inform the application about events of interest.

The following argument variables are passed to the event channel:

SOURCE	The type of the source object, a text keyword
SOURCE_NR	The identifying number of the source object, integer
EVENT	The event that caused the activation, a text keyword
RT	Event time, time data
RM	Milliseconds of the event time, integer
RQ	Qualified event time, list

The implemented values for SOURCE are "UN", "APL", "HOST" and "IMAGE".

11.3.1.16 Unit events

In the case of unit events (SOURCE = "UN"), the argument variables have the following meaning:

SOURCE	"UN"	
SOURCE_NR	Unit number (logical station number)	
EVENT	"SUSPENDED"	The connection to the station is lost
	"RUNNING"	The connection to the station is re-established
	"MISSING"	The connection to the host station cannot be established, because of a mismatch in STA object configuration between the image and the host.
	"LOST"	The connection to the host station is lost, because the mirroring configuration (either MR or IS) in the host has been changed.
	"FOUND"	The connection to the host station is established, because the mirroring configuration in the host has been changed.
	"HOST_LOST"	The connection to the host application is lost.
	"HOST_FOUND"	The connection to the host application is established.
	"PRIMARY"	A proxy station switched to use the primary routing.
	"SECONDARY"	A proxy station switched to use the secondary routing.
	"FORCED_PRIMARY"	A proxy station switched to use the primary routing by SCIL (STA:BAS set to 1).
	"FORCED_SECONDARY"	A proxy station switched to use the secondary routing by SCIL (STA:BAS set to 2).
	"UNFORCED"	Automatic routing of a proxy station re-enabled by SCIL (STA:BAS set to 0).

The events "RUNNING" and "SUSPENDED" are reported by all station types only if the SE attribute of the NET node (NODn:SSE) is 3 or 4. Otherwise, they are reported only by some station types (for example, "REX"). For more information, see the System Objects manual.

11.3.1.17 APL-APL diagnostic events

The APL-APL diagnostic events (SOURCE = "APL") are events that inform about state changes of external applications. The argument variables have the following meaning:

SOURCE	"APL"	
SOURCE_NR	Application number of the external application	
EVENT	"FOUND"	The connection to the external application is established.
	"LOST"	The connection to the external application is lost.
	"AS_CHANGED"	The state (AS attribute) of the external application has changed.
	"SS_CHANGED"	The shadowing state (SS attribute) of the external application has changed.
	"SP_CHANGED"	The shadowing phase (SP attribute) of the external application has changed.

The following three attributes are reported only when the connection to the external base system has been established (EVENT is other than "LOST").

AS	"COLD", "WARM" or "HOT"	The state of the external application.
SS	"WARM_SEND", "HOT_SEND", "RECEIVE" or "NONE"	The shadowing state of the external application.
SP	"TO_WARM_SD", "WARM_SD", "TO_HOT_SD", "HOT_SD", "TO_WARM_RC", "WARM_RC", "TO_HOT_RC", "HOT_RC" or "NONE"	The shadowing phase of the external application.

11.3.1.18 Host events

The host events (SOURCE = "HOST") are events that inform the image application about the mirroring connections to host applications. The argument variables have the following meaning:

SOURCE	"HOST"	
SOURCE_NR	Application number of the host	
EVENT	"CONNECTED"	The connection to the host is established.
	"LOST"	The connection to the host is lost.
	"DISCONNECTED"	The connection to the host has been disconnected by the image, because there are no stations connected to the host any more.
	"RECONNECTED"	The connection to the host is re-established without losing any events.
	"OVERFLOW"	The event buffer of the host has overflowed. Events have been lost.
	"DOWN"	The connection to the host has been disconnected by the host, because the host application is doing shutdown.

Table continues on next page

"DISABLED"	The communication with the host has been disabled by setting APL:BHE to 0.
"ENABLED"	The communication with the host has been enabled by setting APL:BHE to 1.
"HOST_DISABLED"	The communication with the host has been disabled by the host (by setting APL:BIE to 0).
"HOST_ENABLED"	The communication with the host has been enabled by the host (by setting APL:BIE to 1).

11.3.1.19 Image events

The image events (SOURCE = "IMAGE") are events that inform the host application about the mirroring connections to image applications. The argument variables have the following meaning:

SOURCE	"IMAGE"	
SOURCE_NR	Application number of the image	
EVENT		
	"CONNECTED"	The connection to the image is established.
	"LOST"	The connection to the image is lost.
	"DISCONNECTED"	The image application has disconnected the mirroring session.
	"RECONNECTED"	The connection to the image is re-established without losing any events.
	"OVERFLOW"	The event buffer for the image has overflowed. Events have been lost.
	"BLOCKING"	The event buffer for the image have become full, but because of the defined event queue overflow policy "KEEP", the buffer is not discarded. The connection is now blocking (or slowing down) the communication between the SYS and the NET in order to not to lose any events in the image application.
	"NON_BLOCKING"	The event buffer for the image is not full any more. The connection does not slow down the communication between the SYS and the NET. This event is generated when the length of the event queue (EU) has dropped below 90 % of its maximum (EM).
	"DISABLED"	The communication with the image has been disabled by setting APL:BIE to 0.
	"ENABLED"	The communication with the image has been enabled by setting APL:BIE to 1.
	"IMAGE_DISABLED"	The communication with the image has been disabled by the image (by setting APL:BHE to 0).
	"IMAGE_ENABLED"	The communication with the image has been enabled by the image (by setting APL:BHE to 1).

11.3.2 Predefined system event channels

11.3.2.1 IP_EVENT

Event channel IP_EVENT is activated whenever the state of an Integrated Program (IP) changes. This event channel is implemented to supervise the IP's of the system, for example to generate an alarm when an IP becomes non-responsive.

The following argument variables are passed to the event channel:

IPN	The (SCIL) number of the IP
EVENT	Keyword describing the event:
"START"	The IP has been started.
"STOP"	The IP has stopped.
"CRASHED"	The IP has terminated abnormally.
"NO_RESPONSE"	The IP no longer responds.
"RECOVERED"	The IP has recovered from the NO_RESPONSE state.
TIME	The time stamp of the event (qualified time)
START_COMMAND	The IP_START argument used to start the IP
ARGUMENT	The SCIL argument passed to the IP
PROCESS_NAME	The name of the Windows process running as an IP.
PROCESS_ID	The process id (PID) of the Windows process.

11.3.2.2 Generic system event channel

SYS_EVENT is a generic predefined event channel used to inform the application(s) about various system level events of interest.

Each application of the system may choose whether it wants to respond to these events or not. If the EE attribute of the application object (APLn:BEE, see the System Objects manual) is set to 1, the event channel SYS_EVENT is activated in the application, otherwise not. More than one application may receive these system events, but the applications should, in this case, agree upon the reactive actions to the events.

The following argument variables are passed to the event channel

SOURCE	The type of the source object, a text keyword
SOURCE_NR	The identifying number of the source object, integer
EVENT	The event that caused the activation, text
RT	Event time, time data
RM	Milliseconds of the event time, integer
RQ	Qualified event time, list

The possible sources of events as well as the meaning of argument variables SOURCE_NR and EVENT are described below. In each case variables RT, RM and RQ tell the time of the event. Additional argument variables are passed by some event sources.

11.3.2.3 Printer events

The printer supervision done by the base system may generate the following events:

SOURCE	"PRI"
SOURCE_NR	The physical printer number
EVENT	"LOST" Off-line, out of paper, paper jam etc.
	"FOUND" OK after being LOST.
	"OUTPUT LOST" Some output lost due to queue overflow

The "OUTPUT LOST" event is generated when the printer queue length PRIn:BQU exceeds the maximum length PRIn:BQM (see the System Objects manual).

11.3.2.4 Node events

The node diagnostics (see the System Objects manual) may generate the following events:

SOURCE	"NOD"	
SOURCE_NR	The node number	
EVENT	"LOST"	Connection to the node is lost
	"FOUND"	Connection to the node is re-established

11.3.2.5 External clock events

The external clock (type PC31/PC32 radio clock, see the System Objects manual) may generate the following events:

SOURCE	"CLOCK"	
SOURCE_NR	0 (not applicable)	
EVENT	"FREE"	The clock has lost synchronization to the transmitter, the quartz of the card is used instead
	"FOUND"	The clock has re-synchronized to the transmitter
	"LOST"	Invalid data, the clock is lost
	"SUMMER/WINTER TIME CHANGE WARNING"	
		The clock soon switches from daylight saving to standard time or vice versa

11.3.2.6 Application state supervision events

The following system events are generated whenever an application has changed its state (APL:BAS):

SOURCE	"APL_AS"	
SOURCE_NR	Application number	
EVENT	"COLD"	
	"WARM"	
	"HOT"	

The following system events are generated whenever an application has changed its shadowing phase (APL:BSP):

SOURCE	"APL_SP"	
SOURCE_NR	Application number	
EVENT	"NONE"	
	"TO_WARM_SEND"	
	"WARM_SEND"	
	"TO_HOT_SEND"	
	"HOT_SEND"	

Table continues on next page

```
"TO_WARM_RECEIVE"  
"WARM_RECEIVE"  
"TO_HOT_RECEIVE"  
"HOT_RECEIVE"
```

11.3.2.7 Memory pool supervision events

The following system events are generated to enable global memory pool supervision:

SOURCE	"GLOBAL_POOL"	
SOURCE_NR	0 (not applicable)	
EVENT	"OVERFLOW"	A memory allocation request has failed due to insufficient global memory. The system may fail to generate this event because generation of an event itself requires some global memory.

This event is generated only once per system start-up. It may, however, be re-enabled by setting attribute SYS:BME.

The following system events are generated to enable local memory pool supervision:

SOURCE	"PICO_POOLi"	i = Monitor number (1 ... 100)
	"REPR_POOLi"	i = Queue number (1 ... 32)
	"PRIN_POOLi"	i = 1 (process printouts) or i = 2 (report printouts)
SOURCE_NR	Application number	
EVENT	"OVERFLOW"	

These events are generated only once per application start-up. They may, however, be re-enabled by setting the ME attribute of the application object.

11.3.2.8 Application queue supervision events

The following system events are generated to supervise various queues of the application:

SOURCE	"APL_EM"	APL:BEU > APL:BEM
	"APL_QMi"	APL:BQU(i) > APL:BQM(i): i = 1 (time channel queue) i = 2 (event channel queue) i = 3 (parallel queues) i = 4 (delayed execution queue)
	"APL_PMi"	APL:BPU(i) > APL:BPM(i): i = 1 (process printouts) i = 2 (SCIL printouts)
SOURCE_NR	Application number	
EVENT	"EXCEEDED"	

Each of these events is generated only once per application start-up. They may, however, be re-enabled by setting the QE attribute of the application.

11.3.2.9 License events

The following license events report various events related to the violation or expiration of the system license. In addition to the standard SYS_EVENT argument variables, the name of license field is always passed to the event channel in the argument variable FUNCTIONALITY.

SOURCE	"LICENSE"	
SOURCE_NR	0 (not applicable)	
EVENT	"VIOLATION"	An attempt has been made to violate one of the fields of the license. The name of the field is passed in the argument variable FUNCTIONALITY as follows:
	"EXPIRATION"	The license has expired.
	"HW_KEY"	No valid hardware key has been found in the hour after the start-up or the system has run for a week after last detection of a valid key.
	"HSB"	Unlicensed attempt to start shadowing take-over.
	"MIRRORING"	Attempt to create a STA:B object with an unlicensed mirroring role.
	"OPC_SERVER"	OPC client's attempt to connect to the DA server with no license, or an attempt to start an A&E server with no license. The argument variable SERVER tells the server: "DA" or "A&E".
	"APPL_OP_C_SERVERS"	Too many application OPC servers (AOPCS) trying to connect to the system.
	"OPC_DA_CLIENTS"	Attempt to start too many OPC DA clients.
	"OPC_AE_CLIENTS"	Attempt to start too many OPC A&E clients.
	"PROCESS_IO"	Attempt to create a process object when all the licensed objects are already used.
	"HISTORY_DATA"	Attempt to create or modify a non-MO (Memory Only) data object violating the limit of history data records.
	"MO_HISTORY_DATA"	Attempt to create or modify an MO data object violating the limit of history data records.
	"HIS_DATABASES"	Attempt to create or take into use a DATABASE logging profile violating the limit of Historian database connections.
	"HIS_TAGS"	Attempt to connect more process or data objects to Historian database(s) than granted by the license.
	"PROTOCOL"	Attempt to create a NET line when the licensed number of lines with the protocol are already used up. The character position in the PROTOCOL field of the license is passed in the argument variable POSITION.
	"WARNING"	This event is generated by license fields:
	"EXPIRATION"	The license is going to expire in near future. The argument variable DEADLINE tells when: within a "MONTH", "WEEK" or "DAY".

Table continues on next page

	"PROCESS_IO"	90% of licensed process io's have been used.
	"HISTORY_DATA"	90% of licensed non-MO history data records have been used.
	"MO_HISTORY_DATA"	90% of licensed MO history data records have been used.
	"HIS_TAGS"	90 % of licensed Historian tags have been used.
"EXCEEDED"	The license limit defined by FUNCTIONALITY has been exceeded at application start-up. The number of the application is passed in the argument variable APL and the number of excess resources in variable AMOUNT.	
	"PROCESS_IO"	The licensed process io's have been exceeded. The application has started, but communication between its process database and the process has been disabled.
	"HISTORY_DATA"	The licensed non-MO history data records have been exceeded. The application has started, but its time channels have been disabled.
	"MO_HISTORY_DATA"	The licensed MO history data records have been exceeded. The application has started but, its time channels have been disabled.
	"HIS_TAGS"	The licensed Historian tags have been exceeded. The application has started, but the DATABASE logging profile that caused the violation has been taken out of use.
"LOCK_STATUS"	"HW_KEY"	The status of the hardware lock has changed. The new status is given in the argument variable STATUS: either "VALID" or "INVALID".
"KEY_STATUS"	"HW_KEY"	The status of an hardware key has changed. The argument variable KEY contains the id of the key (as given in the license) and variable STATUS contains the new status: either "MATCH", "MISMATCH" or "MISSING".
"RECOVERED"	An application has recovered from its disabled state (see event "EXCEEDED"). The number of the application is passed in the argument variable APL.	
	"PROCESS_IO"	The number of process io's no more exceed the license limit. The communication between the process database and the process has been enabled.
	"HISTORY_DATA"	The number of history data records (MO and non-MO) no more exceed the license limit. The time channels have been enabled.

11.3.3 Operating system event channel

The predefined event channel OS_EVENT receives Windows operating system events (stored in the Windows Event Log) converted into SCIL format by the Operating System Event Handler (OSEH).

Like SYS_EVENT, OS_EVENT is a system event channel, and the application must be enabled by APL:BEE to receive these events.

The following argument variables are passed to the event channel:

RECORD_NR	Integer, the event sequence number supplied by Windows
RT	Time value, the time of event
TYPE	A keyword specifying the type of event: "ERROR" "WARNING" "INFORMATION" "AUDIT_SUCCESS" "AUDIT_FAILURE"
DOMAIN	The domain name of the computer that generated the event
COMPUTER	The name of the computer that generated the event
USER	The name of the active user at the time event was logged (if possible to define)
LOG	A keyword identifying one of the three Windows event logs: "APPLICATION" "SYSTEM" "SECURITY"
SOURCE	The name of the source (application, service, driver, subsystem) that generated the event
CATEGORY	Integer, a source specific category id
EVENT	Integer, a source specific event id, which together with SOURCE identifies the message
MESSAGE	A text vector, the source specific message string. The message may contain several lines.

Section 12 Logging profile objects

This section describes logging profiles and their attributes. It is divided in the following two sections:

[Section 12.1](#) [General](#): The use of logging profiles and their function.

[Section 12.2](#) [Logging profile attributes](#): Logging profile attributes.

12.1 General

12.1.1 Use

Logging profile objects are used to define the connection of SYS600 application database to a Historian database.

A Historian database is a high performance time series database designed to store large amounts of real-time data for later analysis and reporting. See the manuals Historian Configuration and Administration, Historian Operation and Historian Monitor Configuration for more information about Historian databases.

12.1.2 Functionality

The following data from the SYS600 database can be logged to a Historian database:

1. OV (Object Value) attribute of process objects of following types:

- BI (Binary Input)
- BO (Binary Output)
- DB (Double Binary)
- DI (Digital Input)
- DO (Digital Output)
- AI (Analog Input)
- AO (Analog Output)
- PC (Pulse Counter)

2. OV (Object Value) attribute of data objects of value type real or integer.

In addition to the value itself, the time stamp and status are also logged (attributes RT and OS, respectively).

The objects to be logged are specified by linking the object to a logging profile object (attribute GP of process and data objects).

The logging profile objects specify where and how the data is logged.

12.1.3 Different types of logging profile objects

There are three types of logging profiles for different purposes:

- An OBJECT type profile is used to group a set of process and data objects that share a set of logging properties.
- A DATABASE type profile specifies one Historian database and a set of logging properties specific to this particular database.
- A HISTORY type profile specifies how the logged data will be aggregated in the Historian database (averaging, summing up, etc.).

Each logged process and data object has a reference (attribute GP) to one OBJECT type logging profile. The profile defines a few logging properties (Compression Accuracy and Discreteness, see below) and up to 10 logging profile pairs of one DATABASE profile and one HISTORY profile (Storage). Thus, the Storage attribute tells the Historian databases to receive the data and aggregates to be used in each database.

A **DATABASE** type profile defines:

- The location and credentials of the database: attributes Database Address, User Name and Password.
- Patterns for deducing values for various Historian tag properties from SYS600 database attributes. The patterns are given as deterministic, static SDDL (SCIL Data Derivation Language) expressions. See the manual Programming Language SCIL for SDDL details.

Additionally, the profile exposes the state of the database connection (Object Status) and various diagnostics (Diagnostic Counters).

A **HISTORY** type profile contains only one additional attribute (History Collection Templates), which simply lists the Historian history collection templates to be used to aggregate the data.

12.1.4 Storage

The maximum number of logging profiles in an application is 65 535. Logging profile objects are stored in the report database file APL_REPORT.000. When the application is WARM or HOT, the logging profiles along with all their attributes are stored in the global memory pool of SYS600 as well (see the System Objects manual).

12.1.5 Object notation

Logging profile object attributes are accessed in SCIL with the following object notation (see also [Section 4](#)):

name:{application}G{attribute}{(index)}

or

name:{application}G{attribute}{index}

where

'name'	is the name of the logging profile
'application'	is the logical application number
'attribute'	is the attribute name
'index'	is an index or an index range

Logging profiles have no default attribute. Consequently, a logging profile object notation without an attribute name always refers to the entire object.

12.2 Logging profile attributes

12.2.1 Common attributes

12.2.1.1 CM Comment

A freely chosen comment text.

Data type:	Text
Value:	Any Unicode text, up to 255 characters
Access:	No restrictions

12.2.1.2 IU In Use

This attribute indicates whether the logging profile is in use or not. When an OBJECT type logging profile is not in use (IU = 0), the process and data objects connected to the profile are not logged. When a DATABASE type logging profile is not in use, the Historian database is disconnected. When taken into use, the connection is established.

Data type:	Integer	
Value:	0	Out of use
	1	In use
Default value:	0	
Access:	No restrictions	

12.2.1.3 LN Logical Name

The name of the logging profile.

Data type:	Text
Value:	Maximum length 63 characters.
Access:	Read-only, configurable

12.2.1.4 PT Profile Type

The profile type of the logging profile object.

Data type:	Text	
Value:	"OBJECT"	Object profile
	"DATABASE"	Database profile
	"HISTORY"	History collection profile
Access:	Read-only, configurable	

12.2.1.5 ZT Modification Time

The time when the object was created or modified. This attribute is set by the base system when the object is created, and it is updated each time the object is updated by the #MODIFY command (for example, by an object definition tool).

Data type:	Time
Access:	Read only

12.2.2 Object profile attributes

12.2.2.1 CA Compression Accuracy

The compression accuracy used by the objects connected to the profile. Historian database implements a compression algorithm used to save disk space required by the logged values. Compression accuracy defines the maximum loss of accuracy at any moment of time compared to uncompressed data. The value of this attribute is given in units of the logged data (kV, A ...).

Data type:	Real
Value:	Requested compression accuracy. Value 0.0 implies that compression is not used.
Default value:	0.0 (compression not used)
Access:	No restrictions

When CA is changed, all the process and data objects connected to the profile are reconfigured in the Historian database(s).

12.2.2.2 DI Discreteness

The discreteness of the values of objects connected to the profile. Discreteness defines how the trends are drawn in the user interface of Historian. For non-discrete values, any two consecutive logged values are connected by an inclined line. Discrete values are presented as steps (horizontal and vertical lines). Discreteness also affects some aggregate calculations.

Data type:	Text
Value:	"DEFAULT"
	Discreteness depends of the value type of the logged object: Real valued objects are non-discrete, the others are discrete.
	"YES"
	Values are discrete (regardless of the value type).
	"NO"
	Values are non-discrete (regardless of the value type).
Default value:	"DEFAULT"
Access:	No restrictions

When DI is changed, all the process and data objects connected to the profile are reconfigured in the Historian database(s).

12.2.2.3 ST Storage

This attribute specifies the Historian databases where the data from this profile is to be logged and, for each database, how the data is aggregated.

Data type:	Vector of up to 10 lists
Element value:	List of two attributes:
	"DB"

The name of a DATABASE type logging profile that specifies the Historian database to be used.

Table continues on next page

	"HC"	The name of a HISTORY type logging profile that lists the used history collection templates (aggregation).
Default value:	Empty vector	
Access:	No restrictions	

When ST is changed, all the process and data objects connected to the profile are reconfigured in the Historian database(s).

12.2.3 Database profile attributes

12.2.3.1 DA Database Address

The address of the Historian database.

Data type:	Text, up to 255 characters
Value:	DNS or IPV4 address of the database installation
Default value:	""
Access:	Read, conditional write

The attribute can be changed only when the object is out of use (IU = 0).

12.2.3.2 DC Diagnostic Counters

Diagnostic counters of the Historian database connection.

Data type:	Vector of 13 integers:																										
Indexing:	<table> <tr> <td>1</td> <td>Connect count, transitions to CONNECTED state from states NONE or DISCONNECTED</td> </tr> <tr> <td>2</td> <td>Lost count, transitions to state LOST (communication breaks)</td> </tr> <tr> <td>3</td> <td>Reconnect count, transitions from LOST to CONNECTED state (recovered communication breaks)</td> </tr> <tr> <td>4</td> <td>Fail count, unintentional transitions from CONNECTED or LOST to DISCONNECTED state</td> </tr> <tr> <td>5</td> <td>Connect time, seconds since last connect</td> </tr> <tr> <td>6</td> <td>Tag count, number of connected tags</td> </tr> <tr> <td>7</td> <td>Kilo-operations, number of database operations divided by 1000. A counted operation is either a logging of a tag value, or a creation of a tag, or a change in the configuration of a tag.</td> </tr> <tr> <td>8</td> <td>Operations, remainder after division (see above)</td> </tr> <tr> <td>9</td> <td>Throughput, written values per second, floating one minute average</td> </tr> <tr> <td>10</td> <td>Maximum throughput</td> </tr> <tr> <td>11</td> <td>Queue limit, maximum length of the send queue</td> </tr> <tr> <td>12</td> <td>Queue length, current length of the send queue</td> </tr> <tr> <td>13</td> <td>Maximum queue length</td> </tr> </table>	1	Connect count, transitions to CONNECTED state from states NONE or DISCONNECTED	2	Lost count, transitions to state LOST (communication breaks)	3	Reconnect count, transitions from LOST to CONNECTED state (recovered communication breaks)	4	Fail count, unintentional transitions from CONNECTED or LOST to DISCONNECTED state	5	Connect time, seconds since last connect	6	Tag count, number of connected tags	7	Kilo-operations, number of database operations divided by 1000. A counted operation is either a logging of a tag value, or a creation of a tag, or a change in the configuration of a tag.	8	Operations, remainder after division (see above)	9	Throughput, written values per second, floating one minute average	10	Maximum throughput	11	Queue limit, maximum length of the send queue	12	Queue length, current length of the send queue	13	Maximum queue length
1	Connect count, transitions to CONNECTED state from states NONE or DISCONNECTED																										
2	Lost count, transitions to state LOST (communication breaks)																										
3	Reconnect count, transitions from LOST to CONNECTED state (recovered communication breaks)																										
4	Fail count, unintentional transitions from CONNECTED or LOST to DISCONNECTED state																										
5	Connect time, seconds since last connect																										
6	Tag count, number of connected tags																										
7	Kilo-operations, number of database operations divided by 1000. A counted operation is either a logging of a tag value, or a creation of a tag, or a change in the configuration of a tag.																										
8	Operations, remainder after division (see above)																										
9	Throughput, written values per second, floating one minute average																										
10	Maximum throughput																										
11	Queue limit, maximum length of the send queue																										
12	Queue length, current length of the send queue																										
13	Maximum queue length																										
Default value:	All zeroes																										
Access:	Read-only																										

Counters 1 to 4 are cumulative since the application start-up. Counters 5 to 13 are cleared at every successful connect. When the database is not connected, the counters 6 to 13 show the final figures from the latest connection.

12.2.3.3 DD Description Pattern D

Description pattern for data objects.

Data type:	Text, up to 255 characters
Value:	An SDDL expression used to deduce the Description property of the database tag for data objects.
Default value:	"LN"
Access:	No restrictions

When DD is changed, all the data objects connected to the Historian database are reconfigured. Also, any such change in attribute value in a data object that affects the value of the SDDL expression triggers the reconfiguration of the data object.

12.2.3.4 DP Description Pattern P

Description pattern for process objects.

Data type:	Text, up to 255 characters
Value:	An SDDL expression used to deduce the Description property of the database tag for process objects.
Default value:	"if(OI <> "", join(" ", IE), LN + ":" + dec(IX, 0)) + " " + TX" If OI is given, combine OI elements and TX. if OI is not given, combine LN, IX and TX."
Access:	No restrictions

When DP is changed, all the process objects connected to the Historian database are reconfigured. Also, any such change in attribute value in a process object that affects the value of the SDDL expression triggers the reconfiguration of the process object.

12.2.3.5 ED Equipment Path Pattern D

Equipment path pattern for data objects.

Data type:	Text, up to 255 characters
Value:	An SDDL expression used to deduce the Equipment Path property of the database tag for data objects.
Default value:	""
Access:	No restrictions

When ED is changed, all the data objects connected to the Historian database are reconfigured. Also, any such change in attribute value in a data object that affects the value of the SDDL expression triggers the reconfiguration of the data object.

12.2.3.6 EP Equipment Path Pattern P

Equipment path pattern for process objects.

Data type:	Text, up to 255 characters
Value:	An SDDL expression used to deduce the Equipment Path property of the database tag for process objects.
Default value:	"join(".", IE)"
Access:	No restrictions

When EP is changed, all the process objects connected to the Historian database are reconfigured. Also, any such change in attribute value in a process object that affects the value of the SDDL expression triggers the reconfiguration of the process object.

12.2.3.7 ND Name Pattern D

Name pattern for data objects.

Data type:	Text, up to 255 characters
Value:	An SDDL expression used to deduce the Proposed Name property of the database tag for data objects.
Default value:	"LN"
Access:	No restrictions

When ND is changed, all the data objects connected to the Historian database are reconfigured. Also, any such change in attribute value in a data object that affects the value of the SDDL expression triggers the reconfiguration of the data object.

12.2.3.8 NP Name Pattern P

Name pattern for process objects.

Data type:	Text, up to 255 characters
Value:	An SDDL expression used to deduce the Proposed Name property of the database tag for process objects.
Default value:	"LN + ":" + dec(IX, 0)"
Access:	No restrictions

When NP is changed, all the process objects connected to the Historian database are reconfigured. Also, any such change in attribute value in a process object that affects the value of the SDDL expression triggers the reconfiguration of the process object.

12.2.3.9 OS Object Status

The connection status of the Historian database.

Data type:	Text	
Value:	"NONE"	The object is not in use.
	"DISCONNECTED"	The database is not connected either because the connection has failed or the connection has been lost for a long time (buffer overflow).
	"LOST"	The network connection to the database has been lost. The logged values are still buffered to wait for possible reconnection.

Table continues on next page

	"CONNECTED"	The database is connected
Default value:	"NONE"	
Access:	Read-only	

12.2.3.10 PW Password

The password that is used for logging into the Historian database.

Data type:	Text, up to 255 characters
Value:	Password
Default value:	""
Access:	Read, conditional write

The attribute can be changed only when the object is out of use (IU = 0).

12.2.3.11 US User Name

The user name that is used for logging into the Historian database.

Data type:	Text, up to 255 characters
Value:	User name
Default value:	""
Access:	Read, conditional write

The attribute can be changed only when the object is out of use (IU = 0).

12.2.4 History collection profile attributes

12.2.4.1 HC History Collection Templates

Names of history collection templates to be used by the Historian database.

Data type:	Text vector, length up to 10
Value:	Names of Historian history collection templates Each name may be up to 63 characters in length
Default value:	Empty vector
Access:	No restrictions

When HC is changed, all the process and data objects connected to the profile (via an OBJECT type logging profile object) are reconfigured.

Section 13 Event objects

13.1 Use

Event objects facilitate automatic event-driven functions in user interface objects (pictures and Visual SCIL objects). The function to be performed is defined as a SCIL program sequence (using the #ON command, [Section 4.3](#)) or as an event method in Visual SCIL objects.

The event objects are useful, for example, for the following purposes:

- To respond immediately to process events in user interface objects. The process objects of interest are defined with event object activation (the attribute EE is set to 1). The response, that is, a SCIL statement or program, is defined with #ON command or as an event program. This makes cyclic updating with update programs unnecessary. See [Figure 16](#) and the next example.
- To communicate between monitors or between a command procedure and monitors. Any SCIL program may activate an event (using #EXEC command) to notify all open monitors for some significant application defined event.

Events may be used only within an application, they are not sent from an application to another.

13.2 Function

Event objects are activated by process events or by SCIL (see below). The activation of an event object is noted in all monitors belonging to the application. In pictures displayed at the moment, it causes an execution of SCIL sequences defined for the event by means of the #ON command. In Visual SCIL objects, it causes the execution of the event methods specified for the event.

When an #ON command is executed (in a picture or in a Visual SCIL object), its argument #ON block (a SCIL statement) is not executed at once but stored for later use. Whenever the event object (given as the first argument of the #ON command) is executed, the #ON block will be executed. See [Figure 16](#).

The #ON commands are picture and picture function specific. Consequently, the main picture, window pictures and all their picture functions may contain an #ON block for the same event. Correspondingly, several Visual SCIL objects may contain an event method for the same event.

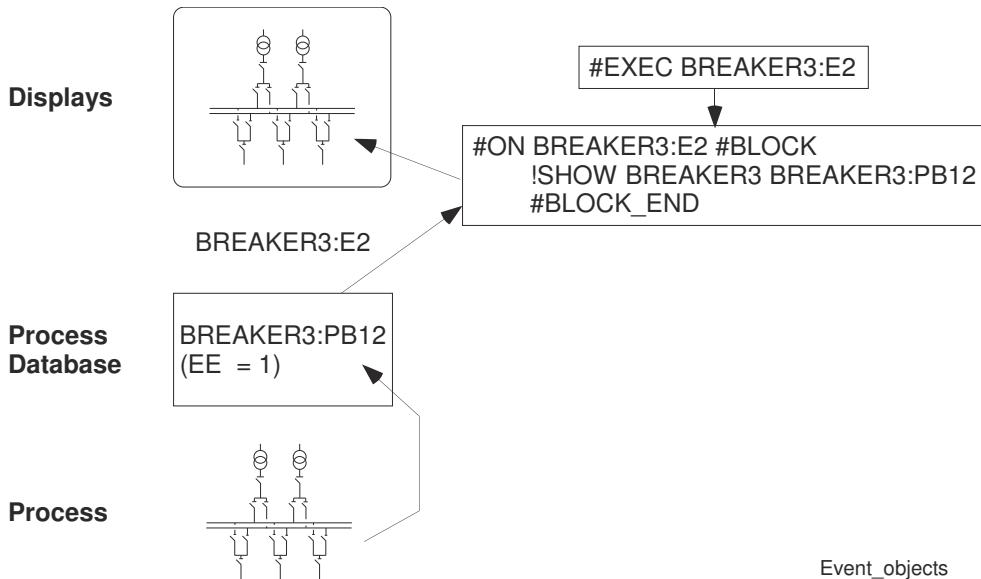


Figure 16: An example that illustrates the use of event objects in pictures.

13.3 Event object execution

As mentioned above, an event object can be executed in two ways:

- **Automatically by the process database.** A change in a process object defined with event object activation (EE = 1, see [Section 4](#)) automatically executes an event object by the same name and index as the process object. The value of the changed attribute is not passed to the event object, nor any information about which attribute has changed. The event object is activated when any of the following attributes is changed:
 - AI, AO, BI, BO, DI, DO, DB, PC, BS, FT, OE, HI, HO, LI, LO, HW, LW, SS, OS, IU, SS, AC, AG, LA, RC, AR, AB, PB, XB, HB, UB, SE, SP, SB, BL, OR, OF, SU, NT
 - For user-defined process object types, other attributes may also be specified to cause an event object execution.
- **By SCIL.** The event objects are activated with the command #EXEC or #EXEC_AFTER ([Section 4.3](#)) In this case, the name of the event object can be chosen freely. Because the event object name space is common to the application, the event name usage should be coordinated by the application.

13.4 Storage

Event objects are not stored.

13.5 Event object notation

The event objects are accessed in SCIL with the following object notation:

name:{application}E{(index)}]

or

name:{application}E{index}

where

- 'name' is the name of the object
- 'application' is the logical application number
- 'index' is an index

Although the syntax allows for the application number, events are not sent from application to another. Consequently, if the application number is given in the notation, it must be the number of the current application.

Event objects have no attributes. They can only be used with the #EXEC, #EXEC_AFTER and #ON commands.

13.6 Example

If the process object

BREAKER3:P2

is equipped with event object execution (BREAKER3:PEE2 = 1), the event object

BREAKER3:E2

is executed each time any of the attributes listed above is changed, see [Figure 16](#).

Suppose a process picture containing the object has executed the following SCIL statement:

```
#ON BREAKER3:E2 !SHOW BREAKER3 BREAKER3:PBI2
```

Now, the position of the breaker is shown in the window BREAKER3 each time the event object is executed.

Section 14 Free type objects

14.1 About this section

This section describes free type objects, their attributes and how to define them. It is divided into the following sections:

- | | |
|------------------------------|---|
| Section 14.2 | General : The use of free type objects and an overview of the attributes. |
| Section 14.3 | Type defining attributes |
| Section 14.4 | Attributes for defining attributes |
| Section 14.5 | Defining free type objects |

14.2 General

14.2.1 Use

Free type objects are used for defining new process object types, user-defined process object types (see [Section 4](#)), and their user-defined attributes. Using free type objects, the programmer can define up to 156 process object types. Each new process object type gets the common process object attributes (see the process object attribute table in [Section 4](#)). In addition, the programmer can design new type specific attributes with desired features. Each user-defined object type has a type number that is used as type definition when creating process objects of the type in question.

Free type objects are mainly used for designing process object types to be used by standard application packages and application extension programs.

14.2.2 Attributes

Free type objects have two types of attributes:

- Elementary attributes that are related to the process object type: name, type number, main attribute (object value attribute), comment text and the total number of user-defined attributes.
- Attributes that define the user-defined attributes: attribute name, data type, automatic activation functions, etc. Each user-defined attribute is defined by a number of free type attributes, which give the user-defined attribute its properties. The user-defined attributes are identified by sequential numbers given as indices.

14.2.3 Object definition

Free type objects are defined with the #CREATE command and deleted with the #DELETE command. A free type object cannot be deleted until all process objects of the corresponding type have been deleted. Individual attributes cannot be deleted. After a free type object has been created, its attributes can be changed with #MODIFY, though with some restrictions:

- The attribute name (AN) is used to identify the attribute. Therefore the attribute name itself may not be changed. A new attribute name is regarded as a new user-defined attribute.
- Due to memory reservation, the new attributes given with #MODIFY do not become valid for the process objects until the application is restarted.

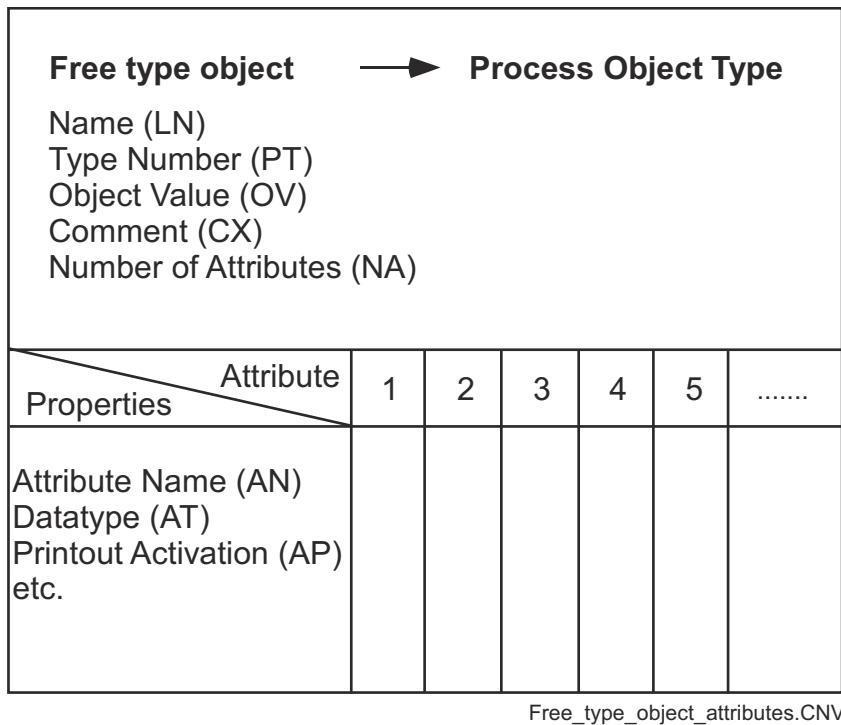


Figure 17: The attributes of free type objects. The type number, that is the PT attribute, is the same as the PT attribute for the process objects. The OV attribute specifies which attribute will be regarded as the OV attribute of the process objects.

The free type objects can also be created using an object definition tool. It can be accessed in the Tool Manager by double-clicking the Free Types icon in the Application Objects page.

14.2.4 Storage

The maximum number of free type objects in an application is 156. Free type objects are stored in the process database file APL PROCES.PRD. When the application is WARM or HOT, the free type objects are stored in the global memory pool of SYS600 as well (see the System Objects manual).

14.2.5 Object notation

Free type objects and their attributes are accessed from SCIL with the following object notation (see also [Section 4](#)):

name:{application}F{attribute}{(index)}

or

name:{application}F{attribute}{index}

where

- 'name' is the name of the object
- 'application' is the logical application number
- 'attribute' is the attribute name
- 'index' is an index or an index range

The index identifies the user-defined process object attribute.

14.3 Type defining attributes

An application can contain up to 156 user-defined process object types. The free type objects have the following five elementary attributes for defining a user-defined process object type:

14.3.1 LN Logical Name

The name of the free type object (and the name of the process object type).

Data type:	Text
Value:	Object name
Access:	Read-only, configurable. Cannot be modified with #MODIFY.

14.3.2 PT Process Object Type

The type number of the process object type.

Data type:	Integer
Value:	100 ... 255
Access:	Read-only, configurable. Cannot be modified with #MODIFY.

14.3.3 OV OV Attribute Name

The name of the attribute that represents the main attribute (the object value). All automatic functions connected to the OV attribute (for example alarm handling, history buffering, automatic printout) will be related to this attribute. The attribute must be of a simple data type.

The attribute OV attribute is not obligatory. If omitted, the automatic OV attribute functions are not performed.

Data type:	Text
Value:	Text of two letters (attribute name), or an empty text string (if no OV attribute).
Access:	No restrictions

Example:

```
#MODIFY TYPE_1:F=list(OV="BB")
```

14.3.4 CX Comment Text

A comment text related to the free type object.

Data type:	Text
Value:	Any Unicode text, up to 255 characters
Access:	No restrictions

14.3.5 NA Number of Attributes

The number of user-defined attributes defined for the type.

Data type:	Integer
Access:	Read-only

14.3.6 ZT Modification Time

The time when the object was created or modified. This attribute set by the base system when the object is created, and it is updated each time the object is updated by the #MODIFY command (for example, by the Free Type Object definition tool).

Data type:	Time
Access:	Read-only

14.4 Attributes for defining attributes

Each user-defined process object type can have a number of type specific user-defined attributes (up to 255). Each user-defined attribute is defined by a name, a data type, and desired activation functions. All features are defined by the F object attributes described below. The attributes are indexed with a sequential number, which identifies the user-defined attributes of the actual type. For example, AN(3) refers to the name of the third attribute of the type.

14.4.1 AN Attribute Name

The name of the attribute. The name can be any two-letter combination not used as a common predefined process attribute name. However, some attribute names imply special functions:

- If the OV value of the object type is integer or real, attributes LW, HW, LI, HI, SZ and AZ will have the same function as the same attributes for analog input process objects.
- If the OV value of the object type is a Boolean value, the AG attribute will have the same function as the AG attribute for binary process objects.

Data type:	Text
Value:	Two character attribute name
Access:	Read-only, configurable. Cannot be modified with #MODIFY, only new user-defined attributes can be appended.

14.4.2 AI Attribute Indexing

The maximum index of the attribute (maximum number of elements) if it is a vector. All elements in the vector are of the same data type, which is given with the AT attribute.

Data type:	Integer
Value:	0 ... 10 000. 0 or 1 means that the attribute is not indexed.
Default:	0
Access:	Read-only, configurable. Cannot be modified with #MODIFY, only new user-defined attributes can be appended.

14.4.3 AT Attribute Value Type

The data type of the attribute, or of the elements of an indexed attribute.

Data type:	Text keyword
Value:	"INTEGER"
	"REAL"
	"TIME"
	"BOOLEAN"
	"TEXT"
	"BIT_STRING"
Access:	Read-only, configurable. Cannot be modified with #MODIFY, only new user-defined attributes can be appended.

14.4.4 AL Attribute Length

The length of the attribute value or the elements of an indexed attribute.

Data type:	Integer	
Value:	The value depends on the AT value (that is the data type of the attribute) in the following way:	
	"INTEGER"	0 Default, the same as 4 (see below)
		4 Signed 32 bit value
		2 Signed 16 bit value
		1 Signed 8 bit value
		-1 Unsigned 8 bit value
		-2 Unsigned 16 bit value
	"TEXT":	(+) n Fixed size of n characters
		- n Variable size, max. n characters where $1 \leq n \leq 255$
	"BIT_STRING":	(+) n Fixed size of n bits
		- n Variable size of max. n bits where $1 \leq n \leq 65\,535$
	For REAL, TIME and BOOLEAN values the attribute is ignored. The length is fixed to 4, 4 and 1 bytes respectively.	
Access:	Read-only, configurable. Cannot be modified with #MODIFY, only new user-defined attributes can be appended.	

14.4.5 AP Attribute Printout

The selection of printout generation at attribute changes. If printout generation is selected, the format picture of the process object (the PF attribute) is printed each time the attribute changes.

Data type:	Integer
Value:	0 No printout generation
	1 Printout generation
Default:	0
Access:	No restrictions

14.4.6 AA Attribute Action

The selection of event channel activation caused by changes of the attribute value. If event channel activation is selected, the event channel of the process object (the AN attribute) is activated each time the attribute is changed.

Data type:	Integer
Value:	0 No event channel activation
	1 Event channel activation
Default value:	0
Access:	No restrictions

14.4.7 AH Attribute History

Selection of history database logging when the attribute is changed.

Data type:	Integer
Value:	0 No history logging
	1 History logging
Default:	0
Access:	No restrictions

14.4.8 AE Attribute Event

Selection of event generation. If event generation is selected, the event object by the name of the process object is activated each time the attribute is changed.

Data type:	Integer
Value:	0 No event generation
	1 Event generation
Default:	0
Access:	No restrictions

14.4.9 AD Attribute on Disk

Updating of the attribute in the process database on disk when the attribute changes.

Data type:	Integer
Value:	0 No updating
	1 Updating
Default:	0
Access:	No restrictions

14.4.10 AS Attribute Snapshot

Defining the attribute as a snapshot variable passed to the event channel and format picture of the process object.

Data type:	Integer
Value:	0 No (the attribute will not be a snapshot variable)
	1 Yes (the attribute will be a snapshot variable)
Default:	0
Access:	No restrictions

14.4.11 AX Attribute Comment Text

A freely formed comment text relating to the attribute.

Data type:	Text
Value:	Any Unicode text, up to 255 characters
Access:	No restrictions

14.4.12 AO Attribute Offset

An informative attribute that tells the starting memory location of the attribute within the consecutive memory area allocated for all the user-defined attributes of the process object.

Data type:	Integer
Value:	0 ... 60 000
Access:	Read-only

Example:

A process object has four user-defined attributes of the following byte length:

1	1 byte	AO(1) = 0
2	2 bytes	AO(2) = 1
3	4 bytes	AO(3) = 3
4	AO(4) = 7

14.5 Defining free type objects

14.5.1 Examples

```
T1:FLN == "T1"
T1:FPT == 101
T1:FOV == "BB"
```

The free type object T1 represents a user-defined process object type with type number 101. The object value (main attribute) of this type is called BB.

Suppose that the BB attribute is the first user-defined attribute of the type (index 1), and that it has the following properties:

```
T1:FAN(1) == "BB"
T1:FAT(1) == "INTEGER"
T1:FAL(1) == 4
T1:FAP(1) == 1
```

This means that the BB attribute is a signed integer of 32 bits. A change in the attribute generates an automatic printout.

Section 15 Variable objects

15.1 Use

Variable objects serve as temporary storage for attributes (attributes gathered from other objects or arbitrary attributes). They are used to form lists, for example, alarm and event lists, to browse through object properties, to copy objects, to create and change objects, etc.

The variable objects have no attributes of their own, but they may be assigned attributes belonging to other object types or arbitrary attributes. The objects as a whole, including all attributes, can be handled as variables of the data type list.



The use of variable objects is more or less obsolete. Because variable objects are essentially global variables with list type contents, accessing them using standard variable access is more powerful and easier to read. See [Section 15.7](#).

15.2 Definition

Variable objects can be created in two ways:

- by #CREATE command
- by assigning a variable a value of a list type expression

Creating a variable object erases the contents of the global variable by the same name.

Using #SET and #MODIFY commands, new attributes are added to the object and values of existing attributes are changed.

Using #DELETE command, the object or one of its attributes is deleted.

15.3 Function

A variable object is, at the same time, both an object and a variable of list type. The list as a whole is handled as a variable, for example %V, while the attributes in the list are accessed with the variable object notation, for example, V:VOV.

Being global variables, the variable objects are accessible only within the SCIL context where they were created.

15.4 Attributes

The attributes of variable objects may have arbitrary names. Unlike other object types, the variable objects can have attribute names containing up to 63 characters. All alpha-numeric characters as well as underscores are allowed in the attribute names, but the first character may not be a digit.

The variable object attributes are used in the same way as other application object attributes. In addition, they can be used in variable expansions (for more information, see the Programming Language SCIL manual).

15.5 Variable object notation

The variable object attributes are accessed with the following object notation:

name:V{attribute}{(index)}

or

name:V{attribute}{index}

where

'name' is the name of the object

'attribute' is the attribute name, up to 63 characters

'index' is an index or an index range

For compatibility to other object types, the index does not have to be surrounded by parentheses if the attribute name consists of exactly two letters. Otherwise, the parentheses are mandatory. In NAME:VAA1, for example, the digit is regarded as an index. In NAME:VAAA1 the digit is regarded as a part of the attribute name. In NAME:VAA(1) the digit is an index.

15.6 Storage

The variable objects are stored as global variables, see the Programming Language SCIL manual.

15.7 Examples

Creating a simple alarm list:

```
#INIT_QUERY "L"
@LIST = PROD_QUERY(20)
!SHOW AT LIST:VAT
!SHOW AM LIST:VAM
!SHOW LN LIST:VLN
!SHOW OX LIST:VOX
!SHOW OV LIST:VOV
!SHOW AR LIST:VAR
```

Creating a data object by copying an existing one:

```
@V = FETCH(0, "D", "DATA1")
#SET V:VLN = "DATA2"
#CREATE DATA2:D = %V
```

The same examples using the global variable notation:

```
#INIT_QUERY "L"
@LIST = PROD_QUERY(20)
!SHOW AT %LIST.AT
!SHOW AM %LIST.AM
!SHOW LN %LIST.LN
!SHOW OX %LIST.OX
!SHOW OV %LIST.OV
!SHOW AR %LIST.AR
```

```
@V = FETCH(0,"D","DATA1")
@V.LN = "DATA2"
#CREATE DATA2:D = %V
```


Section 16 Using object definition tools

This section describes how to access object definition tools through the Object Navigator and how to use the Navigator for application object management (listing, copying, moving, deleting, etc.). It also gives some general principles for using the object definition tools.

16.1 Application Object Navigator

The Application Object Navigator (later Object Navigator) provides the following object handling functions:

- Listing objects of selected types.
- Accessing the object definition forms for viewing and editing.
- Adding new objects.
- Moving an object from one application to another.
- Copying objects within the same application or from one application to another.
- Deleting objects.
- Renaming objects.
- Installing standard functions.
- Documenting application objects.
- Exporting and importing application objects.
- Searching objects.

All application objects are accessible in the Object Navigator. Process display files (files with .v extension) are accessible in the current application, if Object Navigator is started from Monitor Pro, directly or via the SYS600 Tool Manager. Classic Pictures (files with .pic extension) and VSO (Visual SCIL Objects) files (.vso extension) are accessible in local applications and measurement reports in the current application. In addition, contents of representation files can be viewed in local applications.

16.1.1 Entering and exiting object navigator

To enter the Object Navigator, double-click the **Object Navigator** icon in the **Application Object** page of the SYS600 Tool Manager. To exit the Object Navigator, choose **Exit** from the **Object** menu or enter keystroke **CTRL+Q**.

16.1.2 Object navigator composition

The Object Navigator is composed of four main areas, see [Section 16.1.2](#). First are the menu bar and toolbar, where the functions are generally selected. The toolbar itself can be selected from the **Options/Toolbar** menu. Secondly, there is the application and object type tree, where the handled application and object types are selected. Free Type Process Objects, Free Type Objects, Classic Pictures and Classic Representations are shown in the tree if **Options/Obsoletes** has been selected. Thirdly, there is an area for listing object names and indices. Status bar in the bottom of the page shows information on the selected object.

If Process Objects is selected from the object tree, the view options are Process Objects by Groups and Process Objects by Table, otherwise the options are Objects by List and Objects by Table. The view option is selected from the **View** menu.

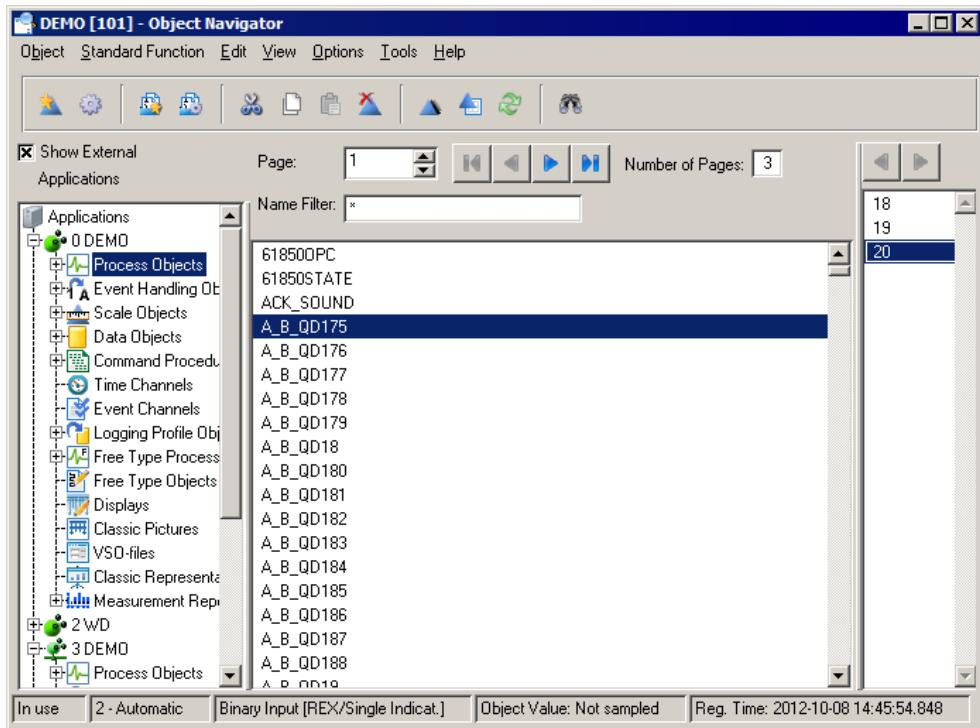


Figure 18: Object Navigator outlook, when process objects are viewed by groups. Status bar shows information on the selected index.

The title of the Object Navigator can be, for example, as follows: DEMO [101] - Object Navigator. The beginning of the title, DEMO, identifies the name of the current application. The [101] stands for the application monitor.

The + or - sign to the left of an application name works as collapse/expand command key. Clicking the + sign makes the object type names visible and accessible, clicking the - sign makes the type names disappear. This makes the list easier to view.

When selecting an object type in an application, the object names of the type in question appear in the list to the right of the object type list. If process objects were selected, the Navigator shows a third list box furthest to the right. This list contains the indices of the selected process object group. By default, all object names, up to one thousand names that meet the filtering criteria of the selected type, are listed.



The buttons with the double arrows, , are enabled if there are more than one thousand names in either direction. Clicking the middle arrow buttons result in displaying the previous or next thousand items. Clicking the leftmost double left arrow button results in displaying the first thousand items. Clicking the rightmost double right arrow button results in displaying the last thousand items or as many as there are left. Another way to browse object names is to use the numeric spinner left to arrow buttons.

The scroll bars allow the user to browse up and down in the lists.

16.1.3 Status bar

The status bar shows information depending on the selected object. See [Table 7](#) and [Table 8](#) below.

Table 7: If the object is selected from object tree, status bar shows this information

Selected object	Information
Application	<ul style="list-style-type: none"> - The type of application (LOCAL, PROXY or EXTERNAL) - State of application (HOT, WARM, COLD or ERROR) - The error text, if the state is error - Computer name, if the type is EXTERNAL - Base system version, if the type is EXTERNAL - The text 'Self-Diagnostics has found conflicts', if conflicts exist
Object type	<ul style="list-style-type: none"> - How many application objects are shown in the table - The total number of application objects - The state of cyclic refresh:on, off or not running

Table 8: If the object is selected from object table, status bar shows this information

Selected object	Information
Process Object Index	<ul style="list-style-type: none"> - In Use or not - Switch State - Object type - Object value - Registration time
Event Handling Object	<ul style="list-style-type: none"> - Event Handling Type - Value Count (of SYS Event Handling) - Value Low (of SYS Event Handling) - Node (of AEC Event Handling) - Event Type (of AEC Event Handling) - Modification time
Scale Object	<ul style="list-style-type: none"> - Scaling algorithm (Linear 1:1, Linear or Stepwise Linear) - Modification time
Data Object	<ul style="list-style-type: none"> - In Use or not - Value Type (REAL, INTEGER, TEXT, TIME) - Comment Text - Numbers of history registration - Registration Time
Command Procedure	<ul style="list-style-type: none"> - In Use or not - Comment Text - Connected Time Channel - Registration Time
Time Channel	<ul style="list-style-type: none"> - In Use or not - Comment Text - Registration Time
Event Channel	<ul style="list-style-type: none"> - Comment Text - Primary Object - The two first Secondary Object(s)
Logging Profile Object	<ul style="list-style-type: none"> - In Use or not - Profile Type - Comment
Free Type Process Object	<ul style="list-style-type: none"> - In Use or not - Switch State - Comment Text - Output Type - Registration Time
Free Type Object	<ul style="list-style-type: none"> - Comment Text - Process Object Type

In [Figure 19](#), the status bar shows that the process object A_E_QD54 with index 10 is in use. The Switch State is manual, the object type is Double Binary Indication (REX/Double Indicat.) and the value is 1. The registration time was 2012-10-08 14:45:55.277.

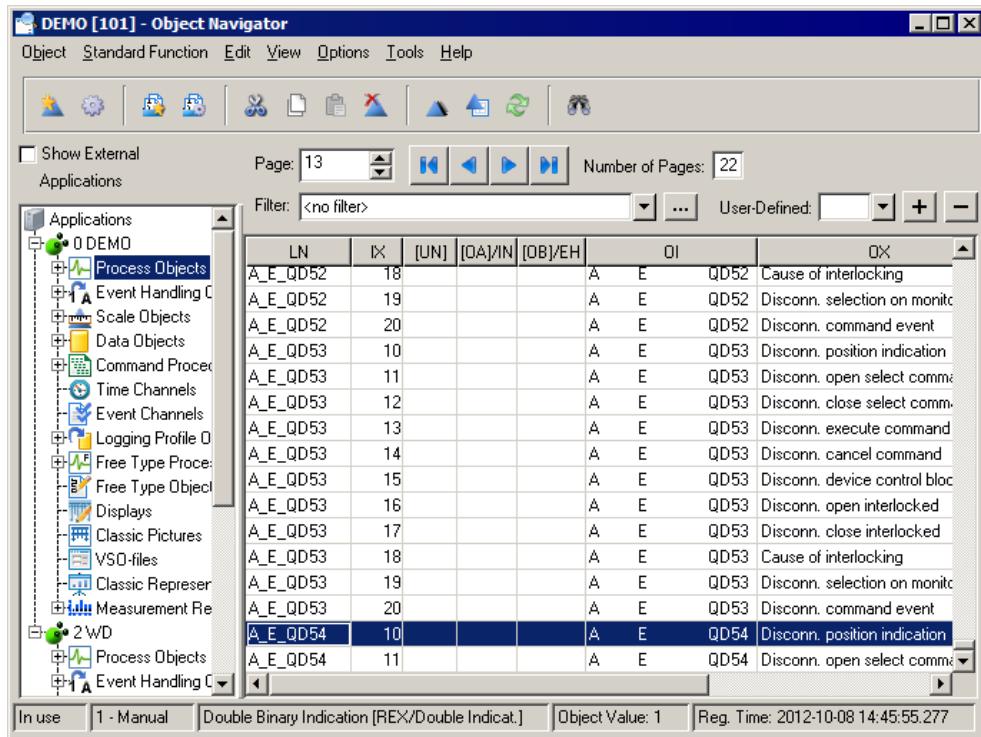


Figure 19: The status bar gives information about the selected process object A_E_QD54

If an application object type is selected from object tree but no objects are selected from the table, the status bar shows the number of objects in view and the total number of objects. For example, when Command Procedures is selected from the Object tree as shown in [Figure 20](#).

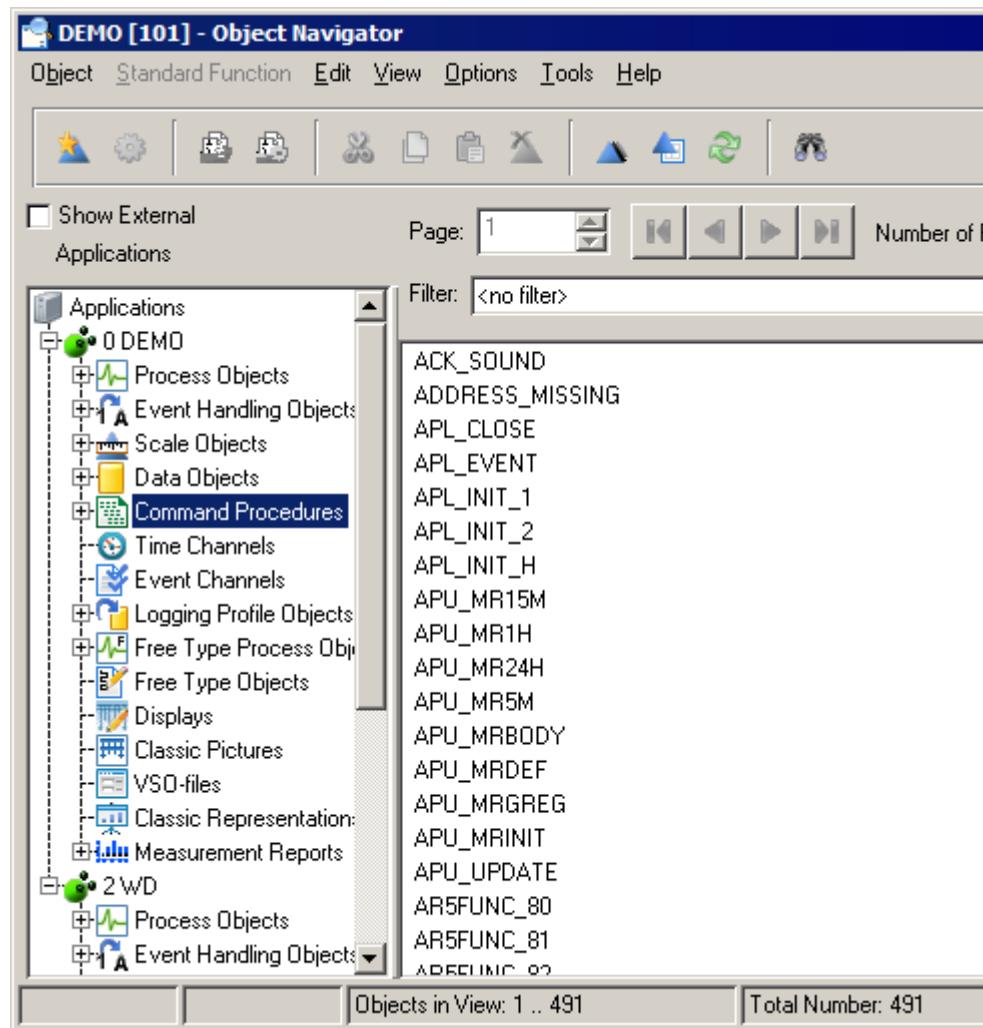


Figure 20: Object Navigator. The status bar shows that the Command Procedures from number one to number 491 or all objects are shown in the list.

16.1.4 Accessible applications

When the Object Navigator is entered, the application and object type tree shows the accessible applications and object types available for the current application. The symbols for the accessible applications are:

- Local application. Green symbol for HOT state and cyan symbol for WARM state applications.
- External application. Only HOT state applications, green symbol, are accessible.

Possible application node types:

- LOCAL HOT: Green symbol.
- LOCAL WARM: Cyan symbol.
- LOCAL COLD: Magenta symbol.
- EXTERNAL HOT: Green symbol.
- EXTERNAL ERROR: Grey symbol.

It is possible to handle objects in the following application types and states only:

- LOCAL HOT.
- LOCAL WARM.
- LOCAL ERROR.
- EXTERNAL HOT.

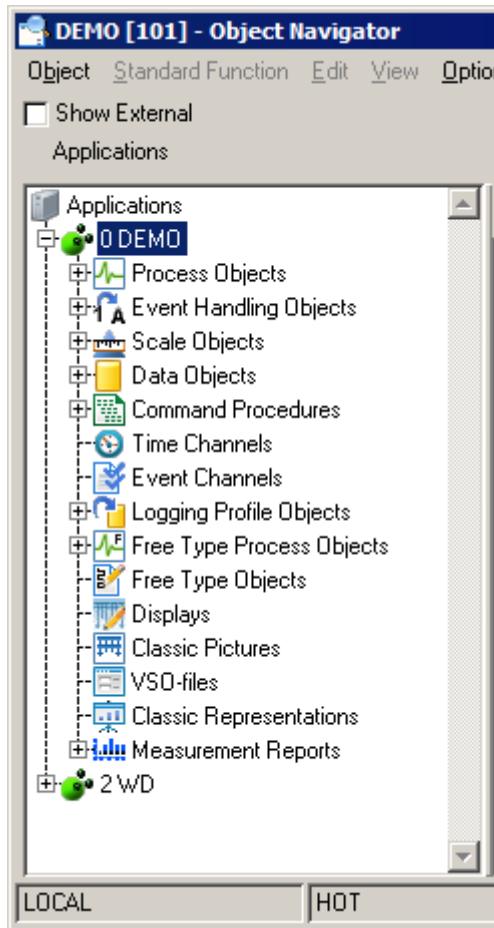


Figure 21: The status bar and the application symbol show that DEMO application is local and hot.

16.1.5 External applications

It is possible to handle objects in external applications in the same way as in local applications, if the mapped applications have MicroSCADA version 8.4.3 or later.

If there are external applications mapped to the current application, the Show External Application check box is enabled as shown in [Figure 22](#). The mapped external applications are shown only when the check box is selected.

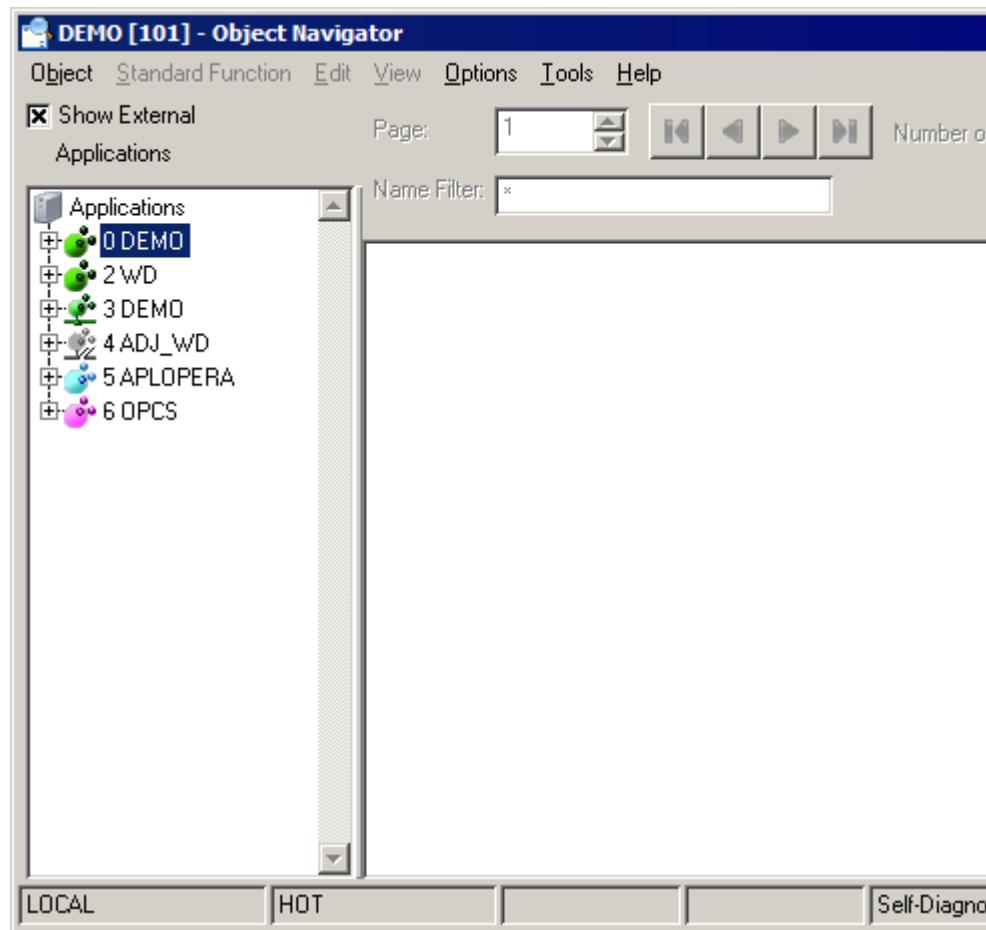


Figure 22: External applications are shown by checking the Show External Applications check box. The local applications DEMO, WD and the external application DEMO are hot.

16.1.6 Proxy applications

A proxy application is shown as local or external, according to its Hot Application.

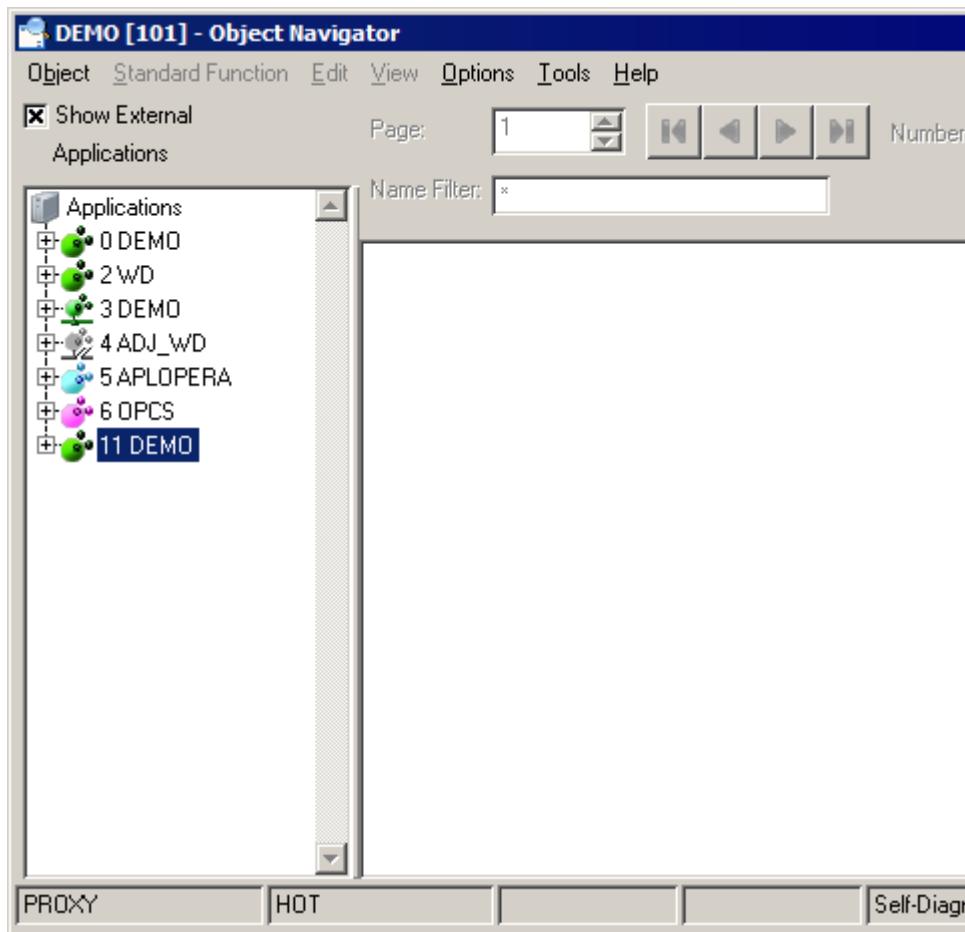


Figure 23: The application with number 11 is a proxy of DEMO application. It is shown in the same way as DEMO application.

16.1.7 Representation options for objects

Application objects have two representation options: the list format and table format. For Process Objects, the list format includes Logical Name (LN) and Index (IX), for other objects only Logical Name (LN). List format is the traditional option, which also shows empty groups for Process Objects. To view the list format, choose **Show/Process Objects by Groups** for Process Objects or for other objects. To view the table format, choose **Show/Process Objects by Table** or **View Objects by Table**. See [Figure 24](#).

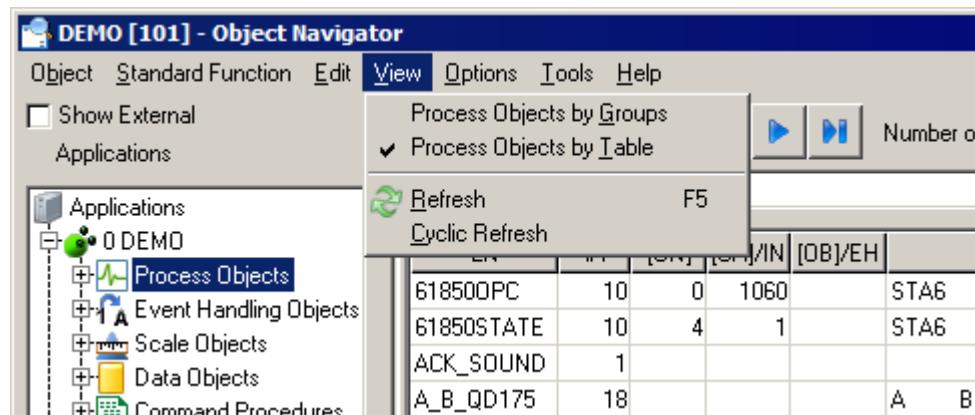


Figure 24: The View menu shows that the representation option at the moment is Table

For table format, the default attributes for Process Objects are LN, IX, UN (Unit Number), OA (Object Address)/IN (Item Name or Source), OB (Object Bit Address)/EH (Event Handling), OI (Object Identifier) and OX (Object Text). The values of OA and OB are encoded depending on object type, but in the list, they are displayed in decimal format. For this reason, they are shown between brackets ([OA]). If all address attributes UN, OA and OB have their initial values as (UN = 0, AO = 0 and OB = 16), the attributes are shown as empty fields. The value of the textual attribute IN instead of OA is shown for objects that use names for signals instead of numeric addresses (objects of type OPC/Binary Input, OPC/Binary Output, ..., OPC/Bit Stream and OPC Event). The value of the textual attribute EH instead of OB is shown for OPC Event objects.

The default attributes for Event Handling Objects are LN and HT (Event Handling Type).

The default attributes for Scale Objects are LN and SA (Scaling Algorithm).

The default attributes for Data Objects are LN, TC (Time Channel) and IN (Instruction).

The default attributes for Command Procedures are LN and TC.

The default attributes for Time Channels are LN, CY (Cycle), SU (Synchronization Unit) and SY (Synchronization Time). Execution part (index 2) of scheduling attributes CY, SU and SY is shown.

The default attributes for Event Channels are LN, OT (Object Type) and ON (Object Name).

The default attributes for Logging Profile Objects are LN and PT (Logging Profile Type).

The default attribute for Free Type Process Objects is LN.

The default attributes for Free Type Objects are LN, PT (Process Object Type), OV (OV Attribute Name) and NA (Number of Attributes).

If a column header is double-clicked in the table, the shown page of the table becomes sorted by the values of that column. If the values of the column can be of any type (cf. OV process object attribute), the values are sorted as they were of text type.

LN is the only obligatory attribute for all objects in the table. Furthermore, the IX attribute is obligatory for process objects. An attribute can be selected from the **User Defined** drop-down list (see [Figure 25](#)). The list includes all the attributes available for the object type, except for attributes of vector and list type. A user-defined attribute is inserted in the table by clicking the button with plus sign. The button with minus sign removes a user-defined attribute from the table. The button removes the last attribute, if no attribute is selected. Note that addressing attributes are selected or removed in one go with ADDR item in the drop-down list. The table can have up to ten attributes.

The screenshot shows a software window with a toolbar at the top containing 'Tools' and 'Help' buttons, and navigation icons. Below the toolbar is a 'Number of Pages:' field set to 22. A 'filter>' input field is followed by a dropdown menu labeled 'User-Defined:' which is currently open, showing a list of attributes: ADDR, AA, AB, AC, AD, AE, AF, AG, AH, and AI. The main area displays a table with columns: IX, [UN], [OA]/IN, [OB]/EH, OI, and Description. Rows 14 through 22 are listed, showing various attributes like QD88, Disconn. cancel, and Disconn. execute command.

	IX	[UN]	[OA]/IN	[OB]/EH	OI	Description
14				A B	QD88	Disconn. cancel
15				A B	QD88	Disconn. devi
16				A B	QD88	Disconn. oper
17				A B	QD88	Disconn. close
18				A B	QD88	Cause of inter
19				A B	QD88	Disconn. selec
20				A B	QD88	Disconn. command
10				A B	QD89	Disconn. position indication
11				A B	QD89	Disconn. open select command
12				A B	QD89	Disconn. close select command
13				A B	QD89	Disconn. execute command

Figure 25: Additional attributes can be selected from the User-Defined drop-down list, if the attributes are shown in table format.

16.1.8 Page length

It is possible to change the page length for the table view by selecting **Object Table Page Length** from the **Options** menu. The **Table Page Length** dialog opens and the number of objects can be selected. See [Figure 26](#).

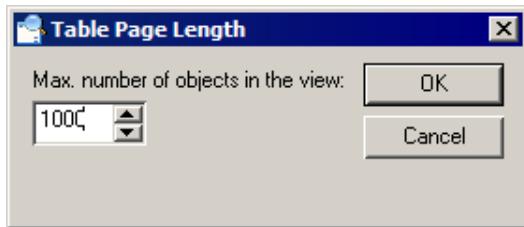


Figure 26: Page length can be defined for application object attributes, if table format is selected.

16.1.9 Auto adjust table columns

It is possible to adjust the last shown column in the table so that it fills the rest of the table when the table is resized or attributes are inserted or removed. The option is set on/off by toggling from the **Options** menu.

16.1.10 Refresh functionality

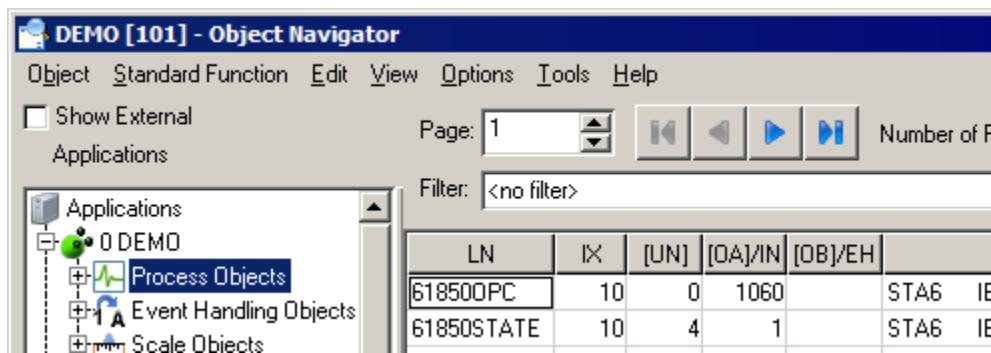
This functionality is needed for updating the view of dynamic attributes when the objects are presented in list format. Refresh function can be carried out by selecting **Show > Refresh** from the menu bar or pressing the F5 key on the keyboard. Cyclic refresh can be set on/off by toggling **Show/Cyclic Refresh** from the menu bar. Cyclic refresh runs only if the number of objects in the list is 100 or less.

16.1.11 Filtering names

To restrict the listed object names, enter a Filter. Use * in the filter to denote one or more characters and % to denote one character in the name. For example, POT* lists objects beginning with a string "POT" and "*" lists all objects, %P* lists all objects, which have P as the second character. All object names that match the filter are listed in the object name list. Name filtering is available for process objects, when the option **Process Objects by Groups** is selected. It is also available for process displays, pictures, VSO-files, representations and measurement reports.

16.1.12 Filtering objects

When viewing objects, an empty filter is used by default. The filters are stored into a parameter file when the Object Navigator is closed, and restored when the tool is opened next time. The last 20 filters can be selected from the drop-down list. A filter can be activated by selecting it from the list. Each application object type has a drop-down list of its own. In [Figure 27](#) the Process Objects are viewed in a table form.



The screenshot shows the 'Object Navigator' window titled 'DEMO [101] - Object Navigator'. The menu bar includes 'Object', 'Standard Function', 'Edit', 'View', 'Options', 'Tools', and 'Help'. There is a checkbox for 'Show External Applications' and a page navigation section with 'Page: 1' and arrows. A 'Filter:' dropdown is set to '<no filter>'. On the left, a tree view shows 'Applications' expanded, with '0 DEMO' selected, revealing 'Process Objects', 'Event Handling Objects', and 'Scale Objects'. On the right, a table displays two rows of process objects:

LN	IX	[UN]	[OA]/IN	[OB]/EH		
61850OPC	10	0	1060		STA6	IE
61850STATE	10	4	1		STA6	IE

Figure 27: Process Objects are viewed in a table form. In this figure there is no filter activated.

16.1.13 Defining a Filter

A new filter can be defined, if **Options/Set Filter...** is selected from the menu bar. This command opens a **Filter** dialog. See [Figure](#). The **Filter** dialog can also be opened from the drop-down list on the right-hand side of the **Filter** field.

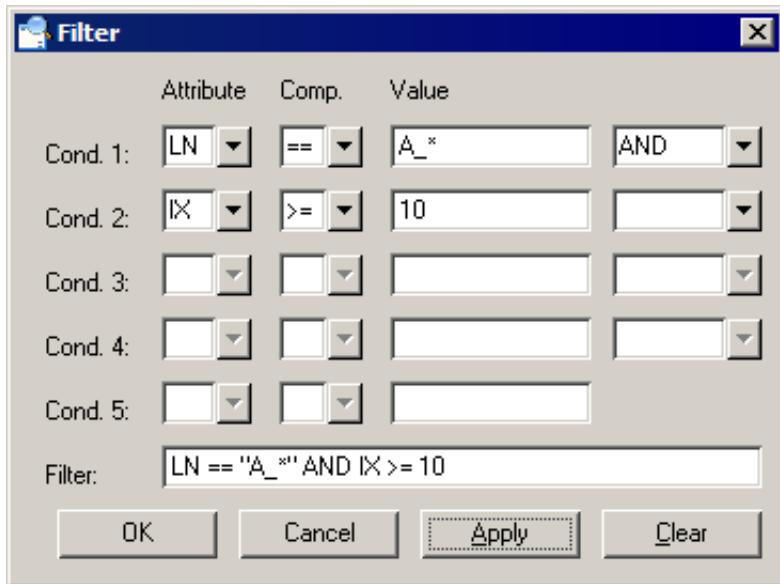


Figure 28: Filter dialog for defining and editing filters. There are two conditions specified in this dialog: LN == A_ AND IX >= 10. The quotation marks are automatically added, if the attribute type is text.*

Only the value should be typed into a value field. If a quotation mark is needed, it is automatically added to the filter.

If there is no filter selected in Object Navigator when the **Filter** dialog is opened, only the first drop-down list is enabled. It is possible to choose any attribute from the **Attribute** drop-down list. After the attribute is selected, the next drop-down list becomes enabled. From this list, it is possible to choose the comparison signs < (smaller than), <= (smaller than or equal to), == (equal to), >= (bigger than or equal to), > (bigger than) or <> (unequal). In the text box, it is possible to type any text. AND or OR has to be chosen from the last drop-down list to be able to enter the next filter condition.

After filling the dialog, the new filter is appended to the drop-down list in the Object Navigator when **OK** or **Apply** is clicked if no SCIL errors were found. If the number of items reaches 20, the last item in the drop-down list is erased.

A validation check is done when **OK** or **Apply** is clicked, or if OR or AND function is chosen. The following message is shown if, for example, text is entered in the first filter condition as an attribute value that should be of integer type.

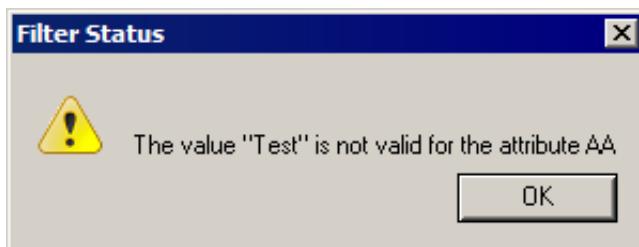


Figure 29: Filter Status info dialog tells that the given attribute value is invalid

OK button updates the contents of process object list in Object Navigator and closes the Filter dialog. **Apply** button refreshes the contents of process object list in Object Navigator, according to defined filter. **Clear** button clears all fields and combo boxes and **Cancel** button closes the dialog without any changes to the Object Navigator.

It is also possible to type the filter condition directly in the **Filter** field. Another check is performed when **OK** or **Apply** is clicked. If there is any SCIL error in the **Filter** field, an error message is shown.

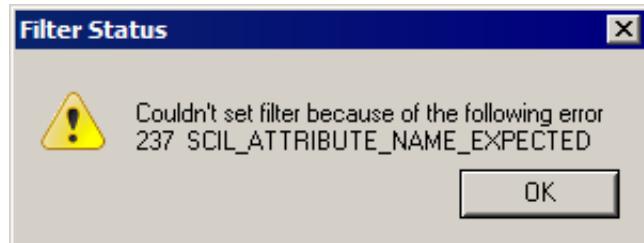


Figure 30: This error message is caused by a missing attribute name in the filter field

16.1.14 Editing attribute values of objects

It is possible to change the attribute value(s) of one or several objects at a time in the **Edit** dialog.

1. From the menu bar, choose **Show/Process Objects by Table**.
2. From the table, select the object(s) to be chosen.
3. Click with secondary mouse button the table and select **Edit** from the pop-up menu.

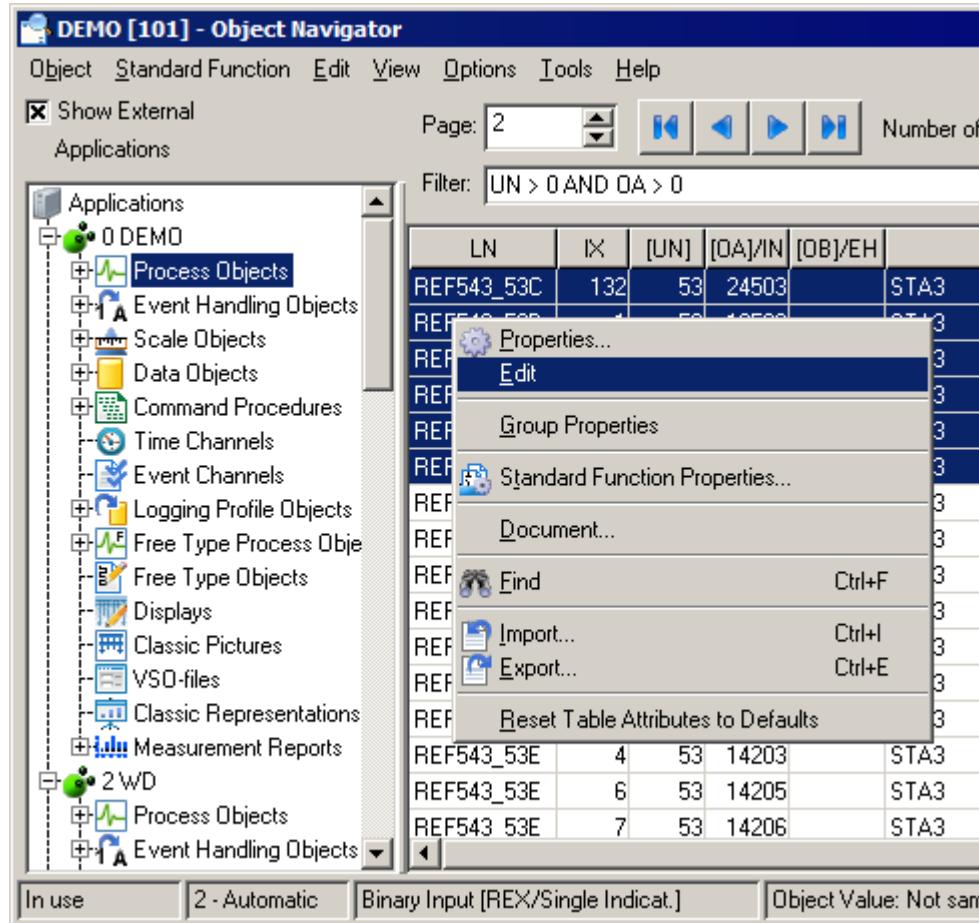


Figure 31: Several objects have been selected from the table and right mouse button has been clicked. To edit the attributes, select **Edit** from the pop-up menu.

The **Edit** dialog opens. It contains the values of the object that is the first one in the selection.

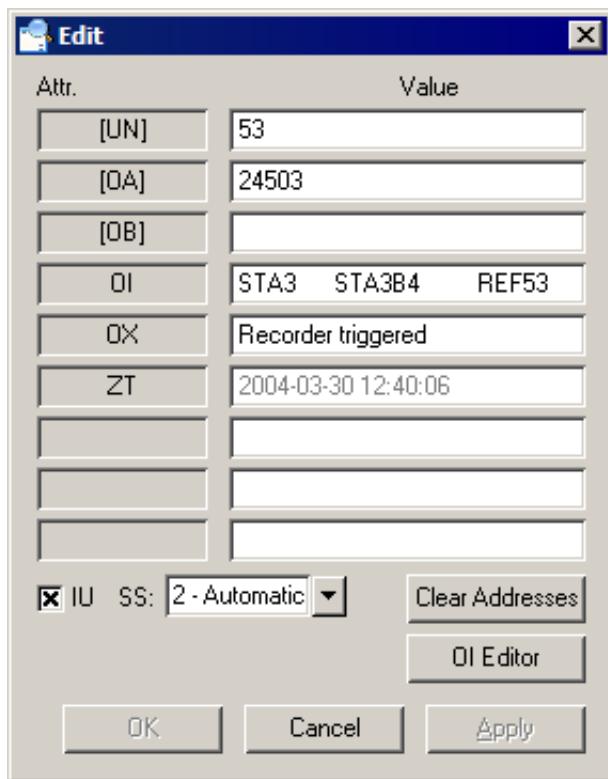


Figure 32: *Edit dialog contains the values of the attribute that was selected first.*

The text fields for OA and OB are disabled if more than one object is selected from the object table. These fields are also disabled if the attributes of the selected object type are not editable.

The **Clear Addresses** button makes it possible to set the address attributes UN, OA and OB or IN to their initial values (UN = 0, OA = 0, OB = 16, IN = "") at once.

Clicking the **OI Editor** button starts the Object Identifier editor. The editor makes it easier to handle a structured Object Identifier attribute. For more information about the editor, see [Section 17.7](#).

The **OK** and **Apply** buttons become enabled after something has been modified. Clicking **Apply** will update the selected object(s) both in the database and in the table. Clicking **OK** will do the same as **Apply** and close the **Edit** dialog.

Error messages are displayed if something goes wrong during the save operation.

16.1.15 Viewing or editing objects

To view or edit the definition of an existing object:

1. Click the object type name below the name of the application where the object is stored. All objects of the selected type appear in the list box to the right. In the case of Process Objects of predefined type, the list shows the names of the process object groups. The object names are listed in alphabetical order.
2. Double-click the name of the object and possibly the index (if the object is a process object of a predefined type), click and choose **Properties** from the **Object** menu, or click the **Object Properties** button in the toolbar.

The object definition tool of the selected object appears and can be edited. Refer to the following sections to learn how to use the object definition tools (see also [Section 14](#)).

16.2 Navigation

16.2.1 Overview

There are six navigation schemes available in the application tree:

- Unit
- Unit and Item Name
- Node and Item Name
- Object Identifier
- OPC Name
- Event Source

The first two schemes and the three last ones are introduced for process objects and free type process objects. The third scheme is introduced for process objects. The fifth scheme is also introduced for data objects and command procedures.

16.2.2 Navigation By Unit

The first categorization, By Unit, is introduced for process objects and free type process objects. In this navigation scheme, the different Unit Number (UN) attribute values found from the selected application become presented. When a Unit Number is selected, all the process objects containing the selected Unit Number value become viewed by Groups or in Table. Unit numbers are sorted in ascending order according to Unit Number attribute value.

If an empty Unit Number value is located, then those process objects become presented under the category <No Address>.

	LN	IX	[UN]	[OA]/IN	[OB]/EH	OI	OX
BNCC2_IESA	10	5	0		NCC 2		IEC104S application comman
BNCC2_IESS	10	5	32000		NCC 2		IEC104S system command h
BNCC2_TRPS	1	5	1		NCC 2		IEC Transparent SPA handlin
BNCC2_TRPS	2	5	2		NCC 2		IEC Transparent SPA handlin
BNCC2_TRPS	3	5	3		NCC 2		IEC Transparent SPA handlin
BNCC_00049	10	5	61204		STA6 BAY2	Q: Disconn. open select comman	
BNCC_00050	10	5	61004		STA6 BAY2	Q: Breaker open select comman	
BNCC_00083	10	5	61104		STA6 BAY2	Q: Disconn. open select comman	
BNCC_00084	10	5	60004		STA6 BAY1	Q: Breaker open select comman	
BNCC_00085	10	5	60104		STA6 BAY1	Q: Disconn. open select comman	
BNCC_00086	10	5	60204		STA6 BAY1	Q: Disconn. open select comman	
BNCC_00097	10	5	31600		STA3 STA3B1	C: Breaker open select comman	
BNCC_00088	10	5	33600		STA3 STA3B3	C: Breaker open select comman	
BNCC_00089	10	5	65004		STA6 BAY6	Q: Breaker open select comman	
BNCC_00090	10	5	41004		STA2 STA2B2	C: Breaker open select comman	
BNCC_00091	10	5	62004		STA6 BAY3	Q: Breaker open select comman	

Figure 33: Navigation by Unit.

16.2.3 Navigation By Unit and Item Name

The second categorization, By Unit and Item Name, is introduced for process objects and free type process objects. In this navigation scheme, the different Unit Number (UN) attribute values found from the selected application become presented. When a Unit Number is selected, all the process objects contained in the unit become viewed by Groups or by Table.

All the different Item Name (IN) attribute values found from the selected unit, become presented. Item Name hierarchy is based on the existence of period (.) in the IN attribute value. For example, if a process object contains the IN value "Stone.IEC.Q1", then it is presented under the category: Stone - IEC - Q1.

Objects that have an empty Item Name are under the category <No Item Name>.

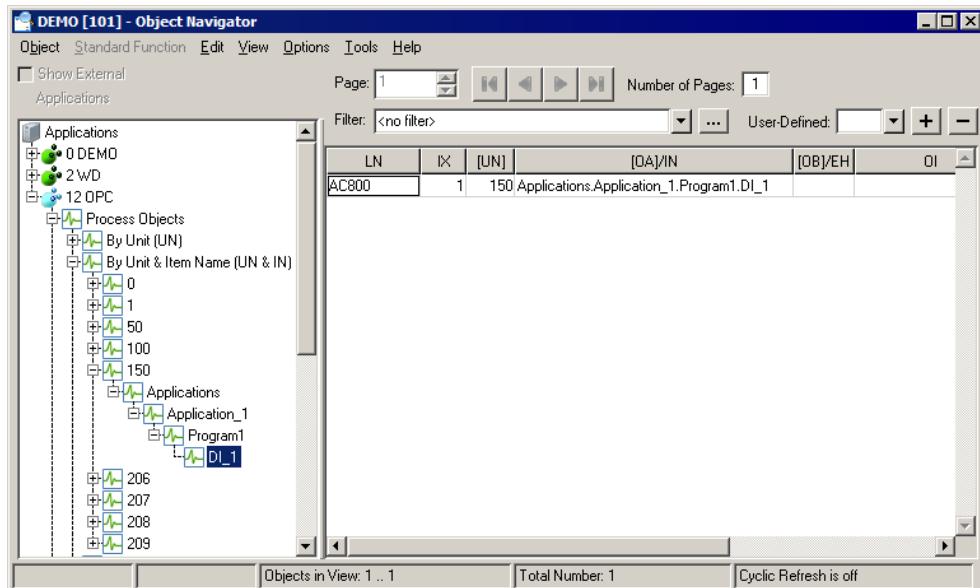


Figure 34: Navigation by Unit and Item Name.

16.2.4 Navigation By Node and Item Name

The third categorization, By Node and Item Name, is introduced for process objects. It is similar to the Unit and Item Name categorization, but in this navigation scheme, the objects are viewed by the base system node they are contained in. When a node is selected, all the process objects contained in a unit that is connected to the node become viewed by Groups or by Table.

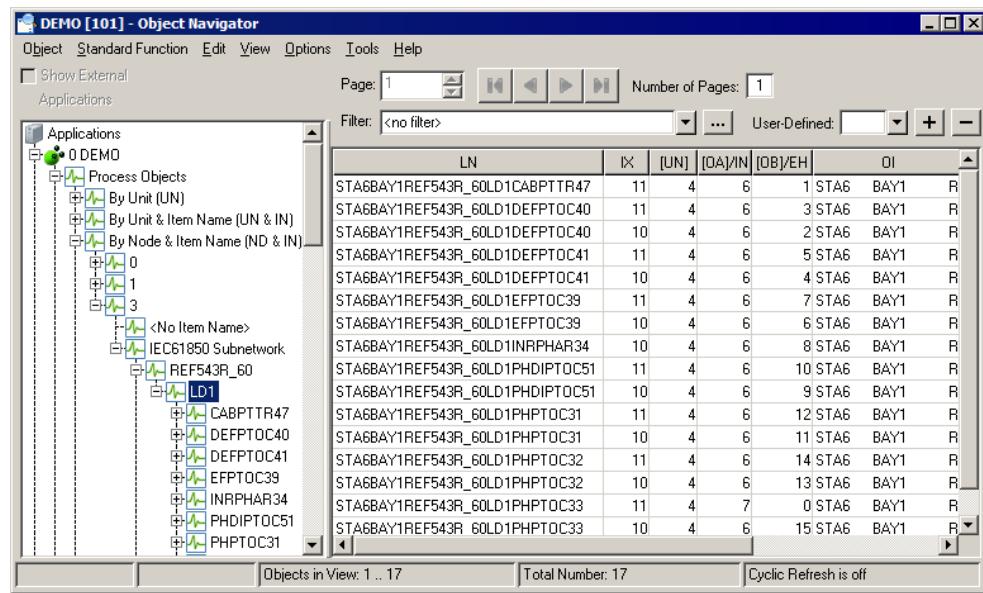


Figure 35: Navigation by Node and Item Name.

16.2.5 Navigation By Object Identifier

The fourth categorization, By Object Identifier, is introduced for process objects and free type process objects. In this navigation scheme, the different Object Identifier (OI) attribute values found from the selected application become presented. During the start-up of Object Navigator tool, the existence of Object Identifier definitions in the application is read and the depth level of Object Identifier is constructed. The depth level may be between 1 and 5. An example of Object Identifier Definition including the depth level 3 is Substation - Bay - Device.

When the first level in Object Identifier category is selected (for example, Substation), all the different Object Identifier belonging to second level (for example, Bay) become listed. Logically this means that all the different Bays belonging to selected Substation become listed.

If an empty Object Identifier value is located, then those process objects become presented under the category <No Object Identifier>.

If the depth level of Object Identifier cannot be constructed at the start-up, there are no categories under the By Object Identifier categorization (except for the category <No Object Identifier>).

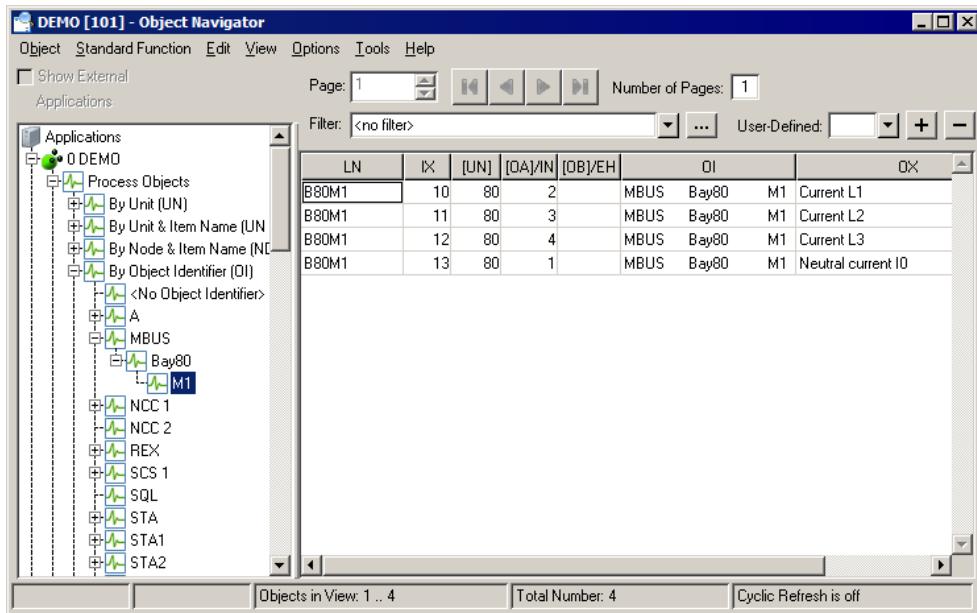


Figure 36: Navigation by Object Identifier.

16.2.6 Navigation By OPC Name

The fifth categorization, By OPC Name, is introduced for process objects, data objects, command procedures and free type process objects. In this navigation scheme, the different OPC Item Name (ON) attribute values found from the selected application become presented. OPC Item Name hierarchy is based on the existence of period (.) in the ON attribute value. For example, if a process object contains the ON value "Eastwick.Incoming 20kV.Q0", then it is presented under the category: Eastwick - Incoming 20kV - Q0

Objects that have an empty OPC Item Name are under the category <No OPC Name>.

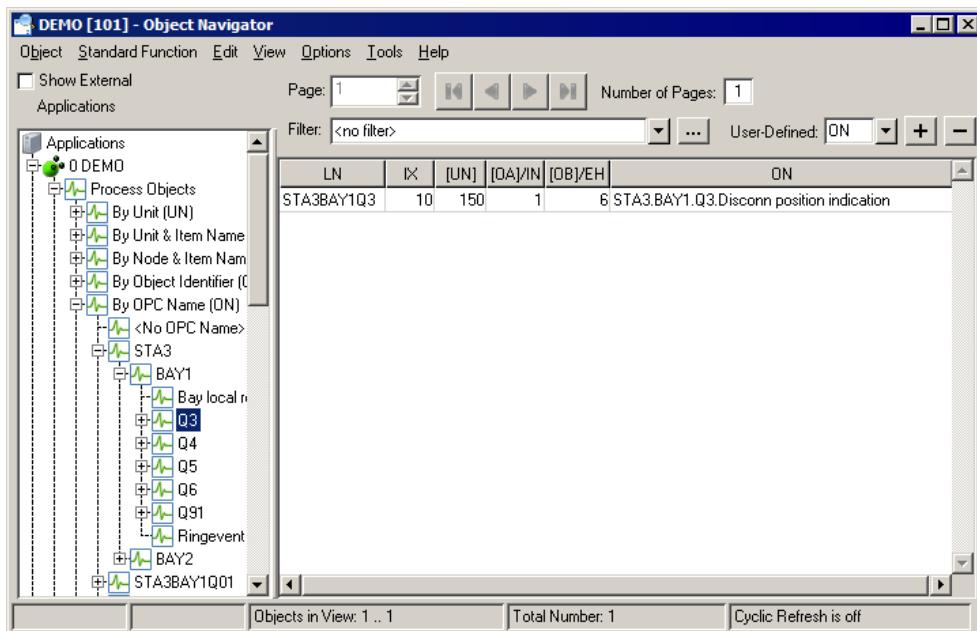


Figure 37: Navigation by OPC Name.

16.2.7 Navigation By Event Source

The sixth categorization, By Event Source, is introduced for process objects and free type process objects. In this navigation scheme, the different Event Source (ES) attribute values found from the selected application become presented. The delimiter in Event Source hierarchy is defined in the OP attribute of the application. For example, if a process object contains the ES value Elgarose.Coles Valley.Q1.Position, it is presented under the category: Elgarose - Coles Valley - Q1 Position.

Objects that have an empty Event Source are under the category <No Event Source>.

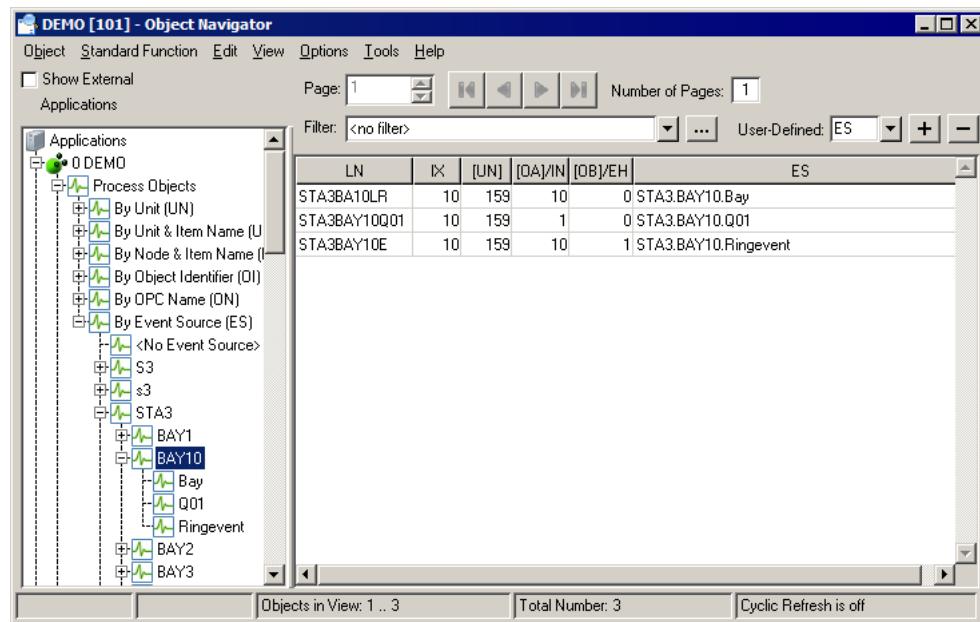


Figure 38: Navigation by Event Source

16.3 Creating and editing objects

16.3.1 Creating new application objects

To create and define a new object in Group or List view:

1. Click the object type in the application where the object will be stored.
2. Select **New** from the **Object** menu, use the shortcut key **CTRL+N** or click **New** button in the toolbar.
3. Type the name in the dialog that appears.
The object (object group for process objects) is created with the given name and the name appears in the object list. The name is the LN attribute of the object.
The object is temporarily located at the beginning of the current page. The given name does not affect the position. The object will be located at the correct position when the object tree is rebuilt, for example, by clicking the base node of the current object type in the tree.
The following step 4 is required only for process object groups.
4. If creating a new process object, index and object type is requested.
5. Select the process object group where the index is created to.
6. Choose **New** from the **Object** menu and input an index in the dialog that appears. Give the index that does not exist under the selected group.
7. Define the process object type and the station type in the dialog that appears (see [Figure 39](#)). It is possible to change the focus between the two list boxes, possible check boxes

and the command buttons with the TAB key. For Analog Input and Analog Output object types, there is a possibility to choose the object representation to be real or 32-bit integer. Default data type is real and the 32-bit integer representation can be chosen by selecting a check box shown in [Figure 40](#). Also, when creating Analog Input and Binary Input DNP, RTU 200 and RTU 200 (EDU) base objects, an option for automatic creation of secondary object is given, see [Figure 40](#). The automatic creation of the secondary object is done during the creation of the base object.

The given index will be the IX attribute of the object and the selected object type will be the PT attribute. The 32-bit integer representation for the Analog Input and Output types will be the IR attribute.

The object has been created with the default values given in Sections 4 .. 12. For some object types, obligatory attributes are assigned values.

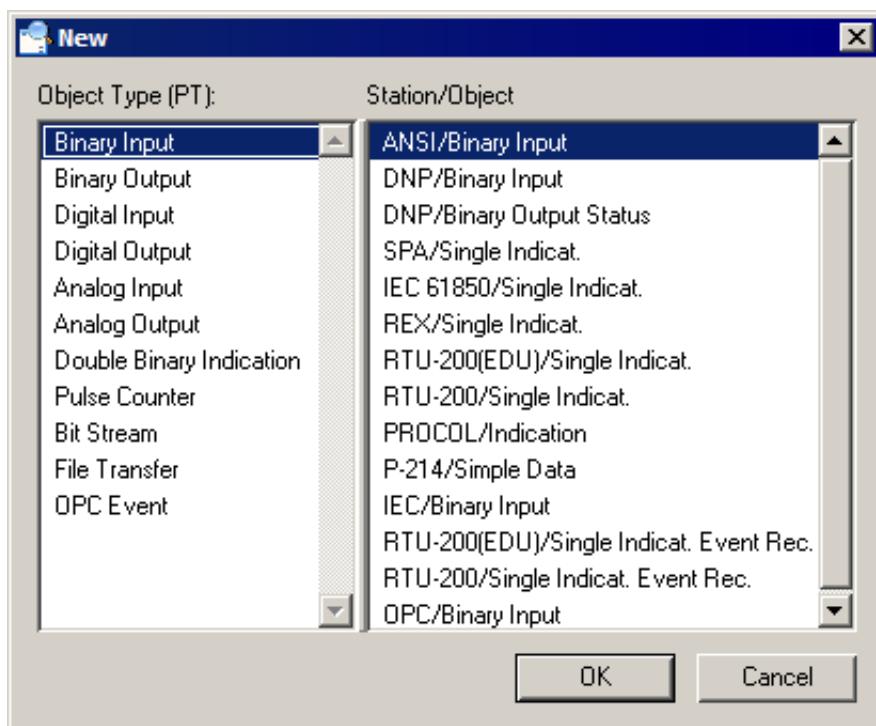


Figure 39: The dialog box where you can choose the object type (PT attribute) of a new process object. Here a binary object will be created.

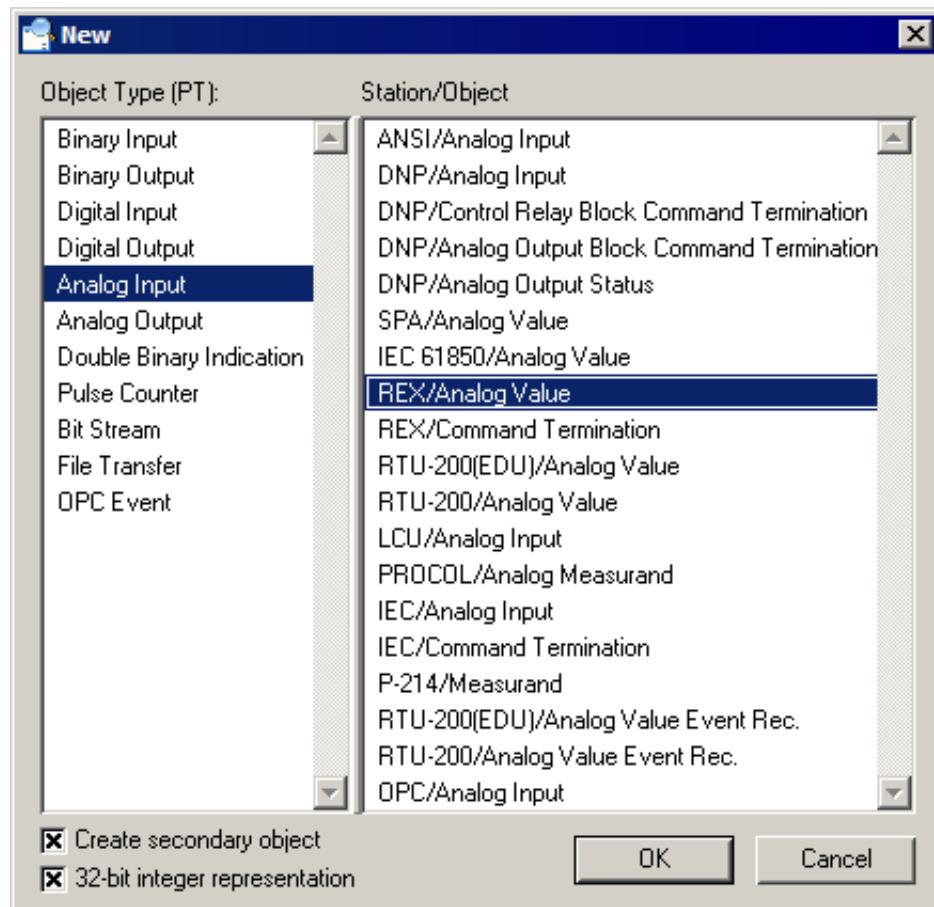


Figure 40: For the Analog Input and Output object types, there is a check box for selecting a 32-bit integer representation. Also for some Analog and Binary Input objects there is check box for automatic creation of secondary object.

16.3.2 Creating new OPC Event objects

To create a new OPC Event process object:

1. Act as described in [Section 16.3.1](#) and select the **OPC Event** object type in **New** dialog.
2. Give the station number (UN attribute) or click the button with three dots and select the number in the dialog that opens.
3. Click **OK**. See [Figure 41](#).

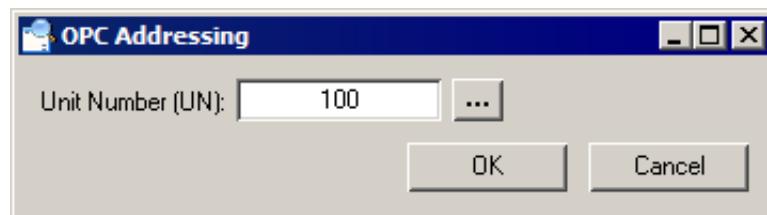


Figure 41: The given or selected value is 100. The station 100 should be of type OAE (OPC Alarms & Events).

16.3.3 Creating new event handling objects

To create a new SYS event handling object to define the event handling of the conventional process objects:

1. Act as described in [Section 16.3.1](#) and select the **SYS** handling type in **New** dialog.
2. Click **OK**. See [Figure 42](#).

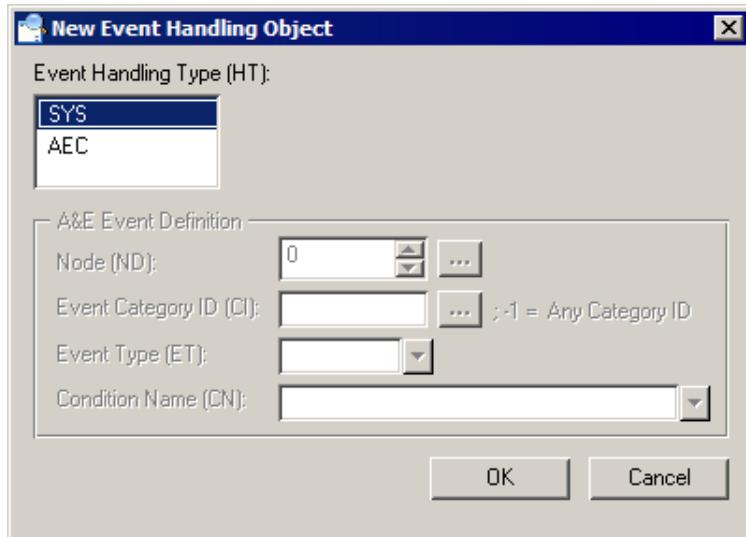


Figure 42: The selected event handling type is SYS.

To create a new AEC event handling object to define the event handling of the OPC Event (OE) type process objects:

1. Act as described in [Section 16.3.1](#) and select the **AEC** event handling type in **New** dialog.
2. Give the node number (ND attribute) or click the button with three dots and select the number in the dialog that opens.
3. Give the event category ID (CI attribute), select the event type (ET attribute) from the drop-down list of options and select the condition (CN attribute) from the drop-down list of options, if the event type is CONDITION or click the button with three dots to load the OPC A&E server event space and select the values of the attributes at once in the dialog that opens.
4. Click **OK**. See [Figure 43](#).

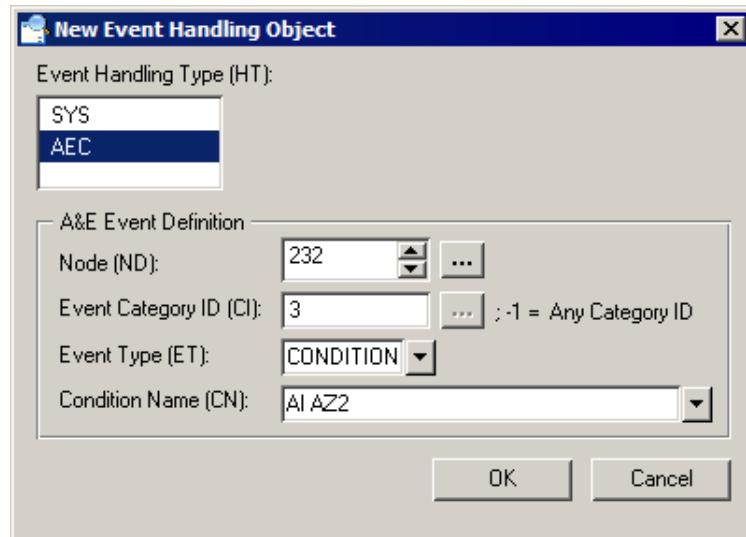


Figure 43: The selected event handling type is AEC.

16.3.4 Creating new data objects

To create a new Data Object:

1. From the menu bar, choose **Object/New**.
2. Give name for the new data object and click **OK**.
3. Give the VT and VL values (VL only if VT is TEXT) in the dialog that opens. See [Figure 44](#).

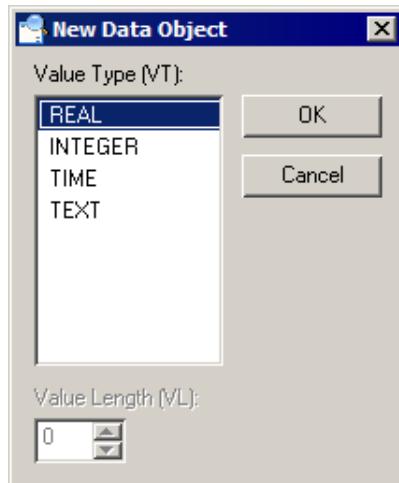


Figure 44: The selected value type is REAL. In this case, it is not possible to give the VL attribute value.

4. Click **OK**.

16.3.5 Creating new logging profile objects

To create a new Logging Profile Object:

1. From the menu bar, choose **Object/New**.
2. Give name for the new logging profile object and click **OK**.
3. Select the PT value in the dialog that opens. See [Figure 45](#).

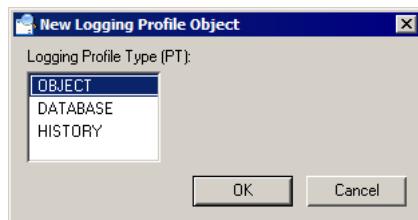


Figure 45: The selected logging profile type is OBJECT.

4. Click OK.

16.3.6 Event recording objects

Event recording objects are process objects created for RTU 200 and RTU 200 (EDU) objects. Their index (IX) should be the index of the supervised object plus 100. The event recording object can be of type BI, AI or DB.

When a new RTU 200 event recording object is created, it gets the Block Address, OB, UN and OI of the supervised object. If the supervised object does not exist, the OA, OB, UN and OI get default values given by the base system. For this reason, the supervised object should be created first.

16.3.7 Defining an application object

To define the application object:

1. Enter the Object Navigator.
2. Select an object type from the application and object tree.
3. Double-click the object name in the list.
4. The object definition tool for the selected object type appears. The attributes have the default values. Define the application object. See the following sections and sections for more information on how to use the tools.

16.3.8 Defining a process object group

To define the process object group:

1. Enter the Object Navigator.
2. Select the Process Objects group type from the application and object tree.
3. Double-click a process object group name in the list.
4. The definition tool for the selected object group appears. The tool is shown in [Figure 46](#). Enter the definitions in it. The name of the logical format picture and names of process views can also be browsed.

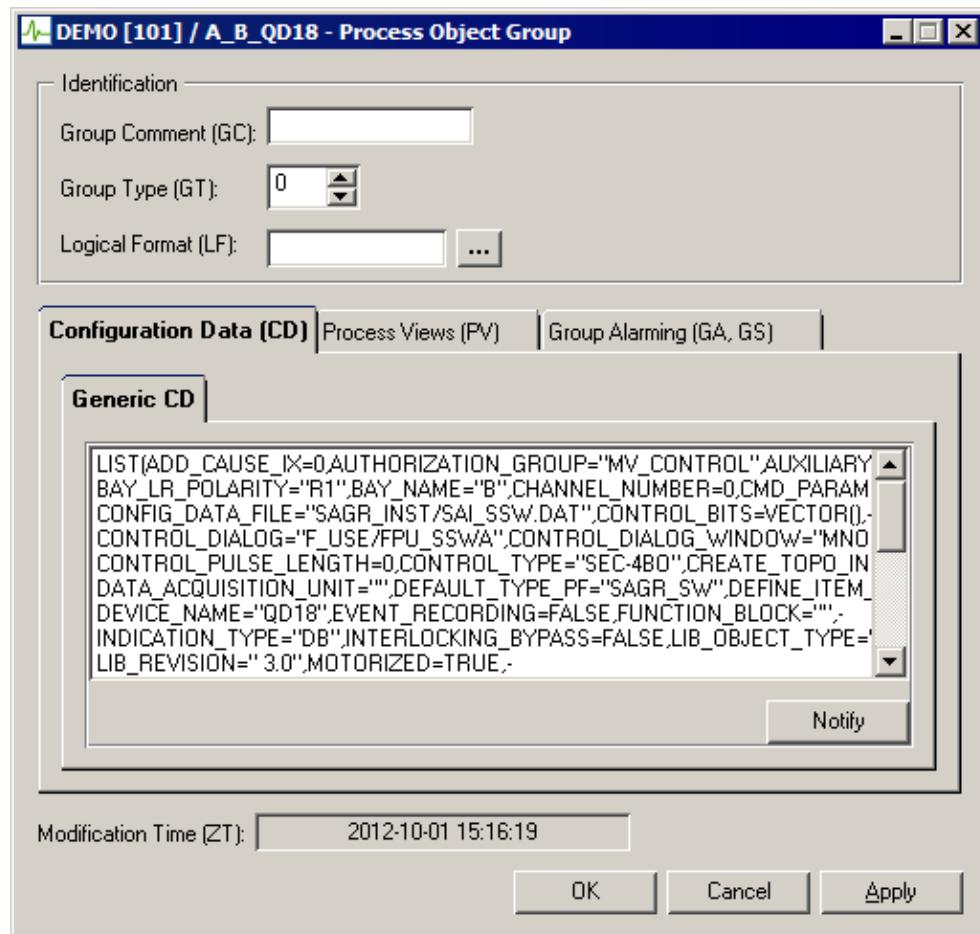


Figure 46: The definition tool for process object group

For process object groups, the group indices have to be defined separately by double-clicking on the index that is to be defined.

16.3.9 Renaming application objects

To change the name of an object:

1. In the left list box, click the application object type that should be renamed.
2. Click the name of the object in the second list box. In the case of process objects, click the process object groups and then the index.
3. Choose **Rename** from the **Object** menu.
4. Type the name of the object in the white text box of the dialog that appears. Click **OK**.

16.3.10 Copying application objects

To quicken application engineering, application objects and their definitions can be copied to create several objects with the same configurations. A maximum of 10 000 objects may be copied per one copy operation, and the same concerns the paste operation as well. Note that the cut operation allows a maximum of 1000 objects to be cut at one time. To copy application objects from one application to another:

1. Click the object type under the name of the application from which objects should be copied.
2. Select an object for copying by clicking the name of the object. Several objects can be selected by holding the CTRL key down while clicking the objects or pressing the mouse button down and dragging the pointer over the objects.
3. Choose **Copy** from the **Edit** menu.
4. Click the object type below the application to which the objects should be copied.
5. Choose **Paste** from the **Edit** menu.

If a Process Object has reference to a Scale Object or Event Handling Object and the referenced objects did not exist before the paste operation in the application where they were pasted to, they will be created there. [Figure 47](#) shows an example of the informative dialog that appears when the referenced Scale and Event Handling Objects are created into an application while pasting. Similarly, if a Data Object or Command Procedure Object has a reference to a Time Channel Object that does not exist, it will be created.

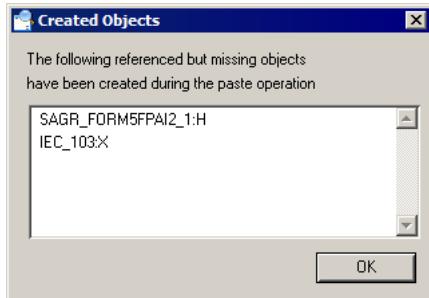


Figure 47: The Created Objects dialog informs that one Event Handling and one Scale Object were created in the application

The copying of objects from a local application to an external application and vice versa is done in the same way as copying between two local applications.

Copied objects are temporarily located at the beginning of the current page. They will be located at correct positions when the object tree is rebuilt.

To copy application objects within an application:

1. Click the object type under the name of the application where objects should be copied.
2. Select an object for copying by clicking the name of the object. Several objects can be selected by holding the CTRL key down while clicking the objects or pressing the mouse button down and dragging the pointer over the objects.
3. Choose **Copy** from the **Edit** menu.
4. Choose **Paste** from the **Edit** menu.

In both cases, the user is reminded of a duplication of object names with the following dialog. The dialog contains six buttons:

Retry	Retries to paste, use if fields have been edited.
Retry to All	Retries to paste the current and all subsequent process objects to the same group. Use if the name field has been edited.
Overwrite	Overwrites existing object.
Overwrite to All	Overwrites the existing object and all subsequent matching objects, if overwrite to all is confirmed by the overwrite prompt (refers to deleting application objects).
Skip	Continues to paste but skips the currently matching object.
Stop	Stops pasting, pasted objects are not cancelled.

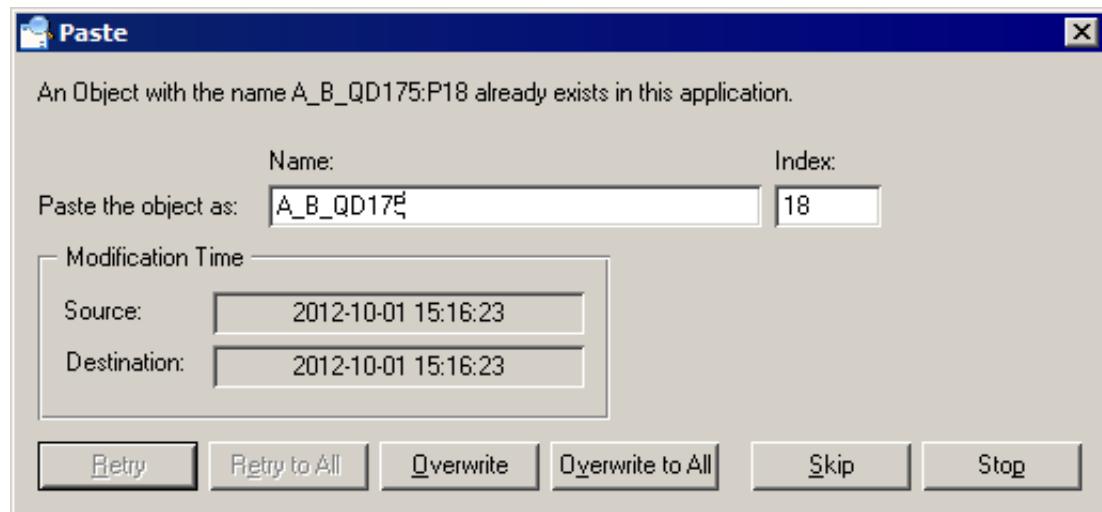


Figure 48: Duplicate object names are not allowed

16.3.11 Copying process objects index

To copy the definition of a process object and to paste the definition to another object index:

1. Select application.
2. Select process object group.
3. Select an index or several indices.
4. Click **Copy** on the **Edit** menu.
5. Select target application.
6. Select target in one of the following manners:

Case 1	Select a process object group whereas the indexes are pasted under that group
Case 2	No group is selected, the indices are pasted under the object group name stored at copy
Case 3	If several object groups are selected, indices are pasted under the first group in the selection.

7. Click **Paste** on the **Edit** menu.

If Address Overlap error occurs when pasting the process object index, the dialog in [Figure 49](#) is shown.

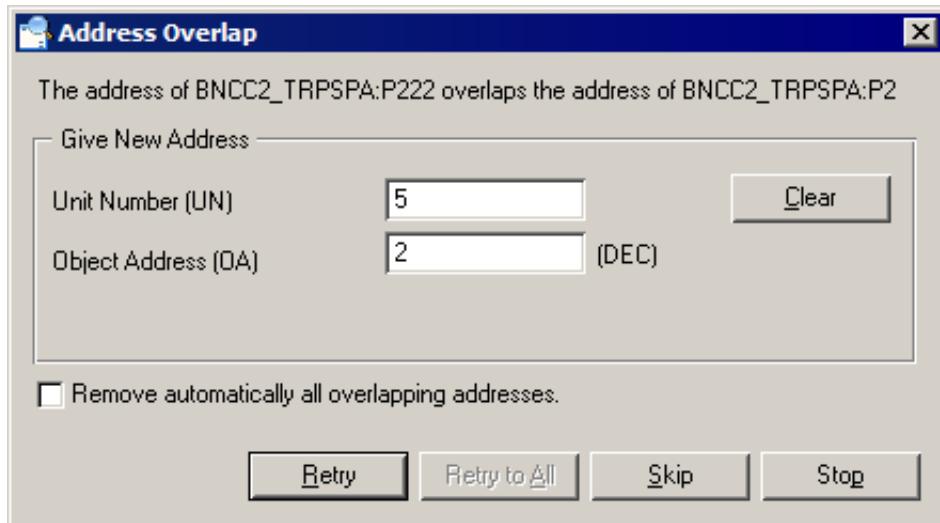


Figure 49: Address Overlap dialog

The user may fill in a new unit number, a new object address or check the "**Remove automatically all overlapping addresses.**" check box. If the check box is selected, all the overlapping addresses of objects that remain to be pasted are cleared after a retry or skip. The addresses that were cleared during pasting are shown in a dialog alongside the corresponding indexes, see [Figure 54](#). Changing the unit number is useful when copying a whole process object group to another station where the object addresses should be the same. The **Retry to All** button is enabled when the unit number is changed. This button makes it possible to change the unit number of the selected process objects at once.

If Address Overlap error occurs when pasting the process object index that uses names as addresses, the dialog in [Figure 50](#) is shown.

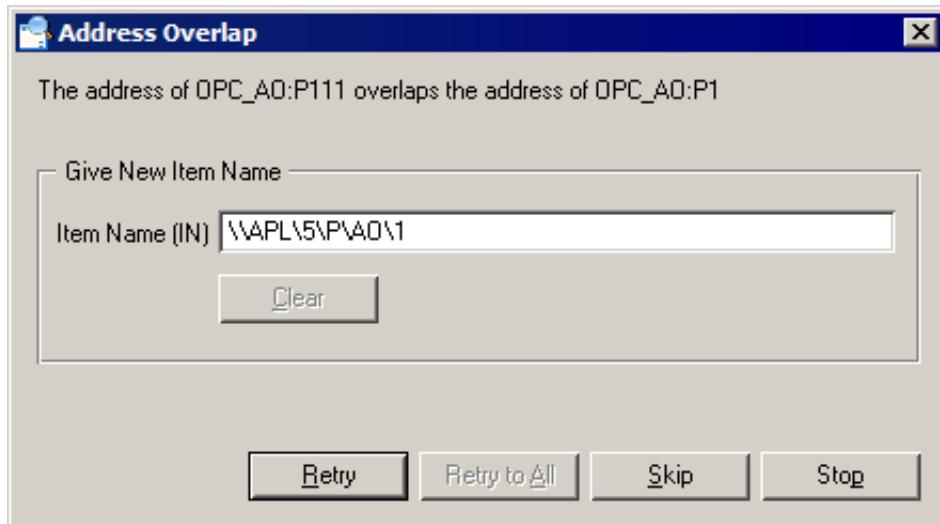


Figure 50: Address Overlap dialog.

If the address conflict occurs when pasting the OPC Event process object index, the dialog in [Figure 51](#) is shown.

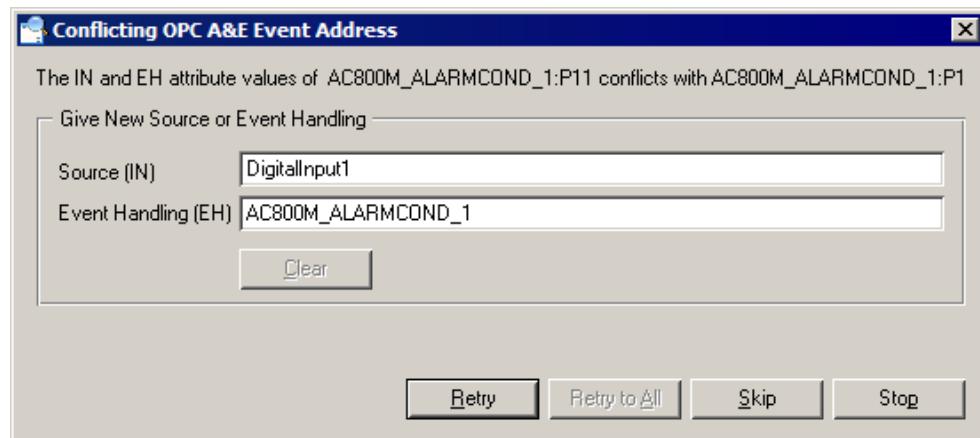


Figure 51: Address Overlap dialog.

If the value of OPC Item Name (ON) attribute conflicts with another object, the dialog in [Figure 52](#) is shown. If the check box is checked, all conflicting ON attribute values of objects that remain to be pasted are cleared after a retry or skip.

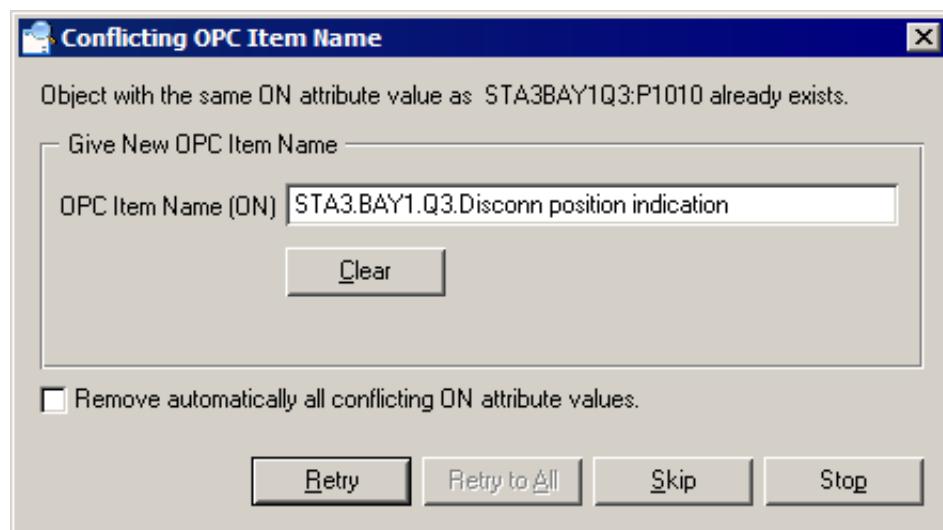


Figure 52: Conflict dialog of ON

If the values of OPC Event Source (ES) and Event Handling (EH) attributes conflict with another object, the dialog in [Figure 53](#) is shown. If the check box is checked, all conflicting ES and EH attribute values of objects that remain to be pasted are cleared after a retry or skip.

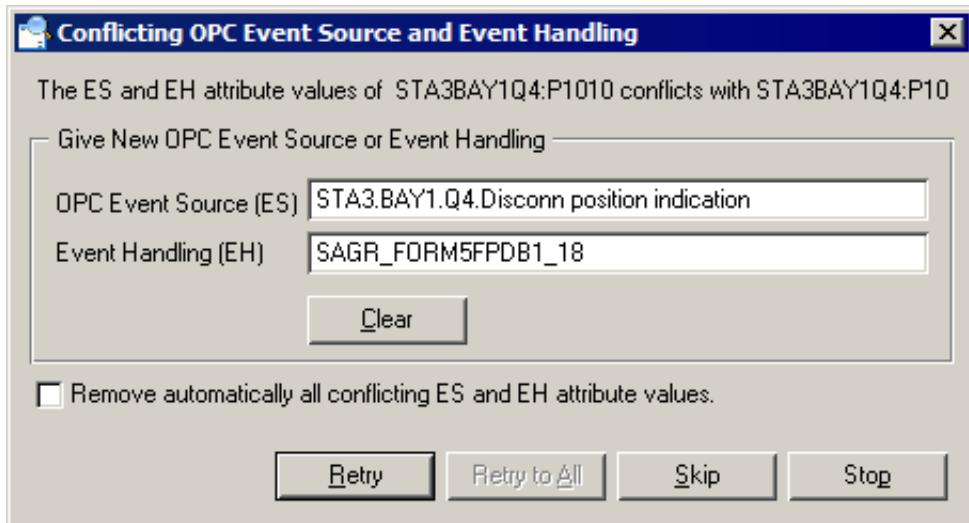


Figure 53: Conflict dialog of ES and EH.

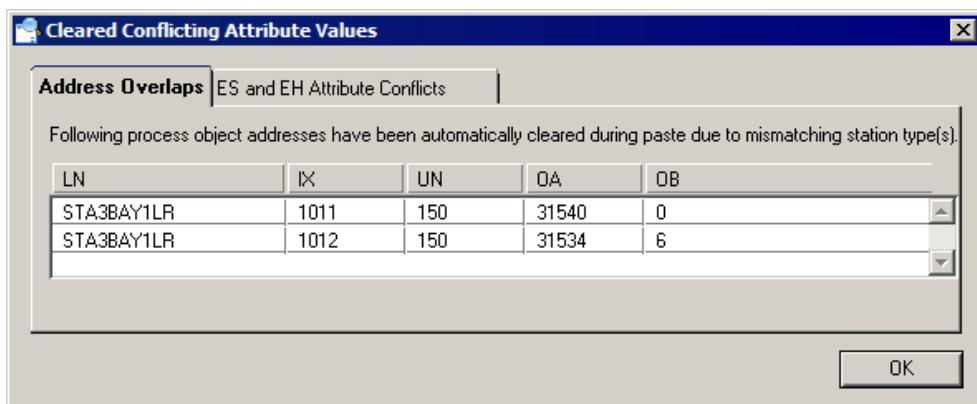


Figure 54: Dialog presenting the addresses that were cleared during operation, alongside the corresponding indexes. Values of ES, EH and ON attributes were cleared too.



Attempt to overwrite a process object index with an index of a different process type (PT) is not allowed. The message "Object can't be overwritten, PT's don't match" is shown. However, if the **Overwrite to All** button is clicked, the check of a process object type is not done.

16.3.12 Copying bay level process objects

To copy the definitions process objects of bay level and to paste the definitions to another objects:

1. Select application.
2. Select By Object Identifier (OI) node in Process Objects node in Navigator's tree and expand it.
3. Go deeper in the tree by expanding nodes and select a bay level node.

4. Click **Copy** on the **Edit** menu or right-click on the tree and select **Copy** from the pop-up menu.
5. Select the node of one level higher than the bay level node. Click **Paste Special** on the **Edit** menu or right-click on the tree and select **Paste Special** from the pop-up menu. Paste Special dialog opens.
6. Edit fields with New ... labels and click **OK** button ([Figure 55](#)).

New process objects including modifications given in the dialog will be created. SA-LIB Configuration Data (CD) attribute changes are also included. The tree is updated with the new bay level node, too.



Copy/Paste Special operation differs from Copy/Paste operation, that no rows are allowed to be selected on the table or group view. To interrupt the operation, click the Cancel button in Paste Special dialog, if opened or click the table or group view.

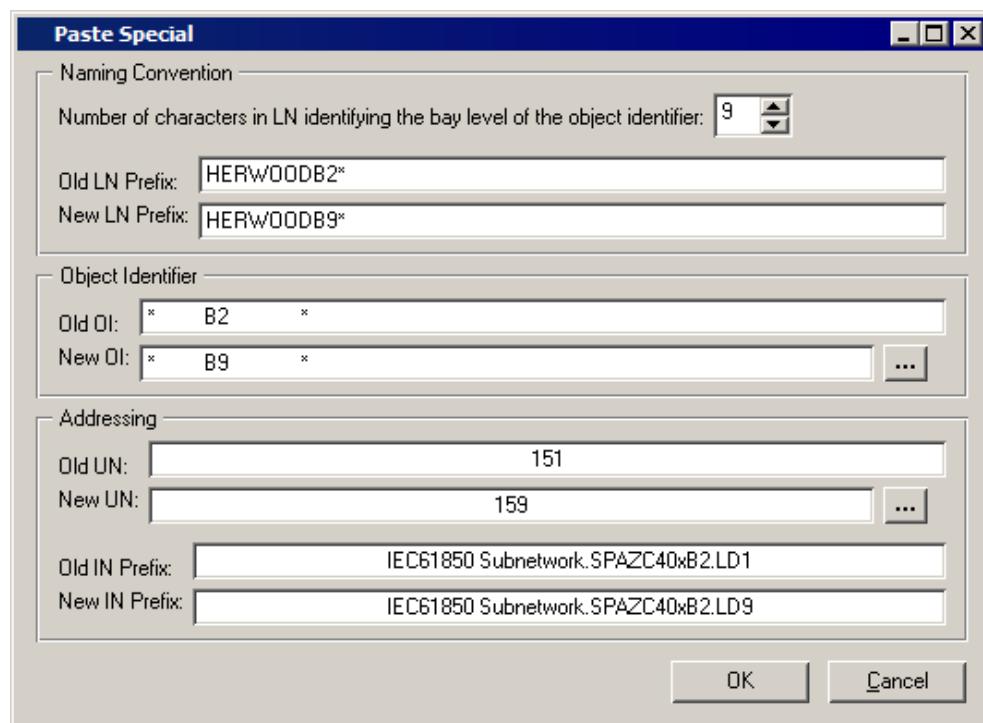


Figure 55: Paste Special dialog

16.3.13 Editing process object identifiers by levels

To edit partly object identifier attribute of process objects in one go:

1. Select application.
2. Select By Object Identifier (OI) node in Process Objects node in Navigator's tree and expand it.
3. Select node or leaf.
4. Right-click on the tree and select **Edit Object Identifier** from the pop-up menu. Object Identifier Editor dialog opens.
5. Edit the enabled field and click the **OK** button ([Figure 56](#)).

Object Identifier attribute of process objects under the node or leaf will be updated including SA-LIB Configuration Data (CD) attribute changes. Navigator's tree is updated, too.

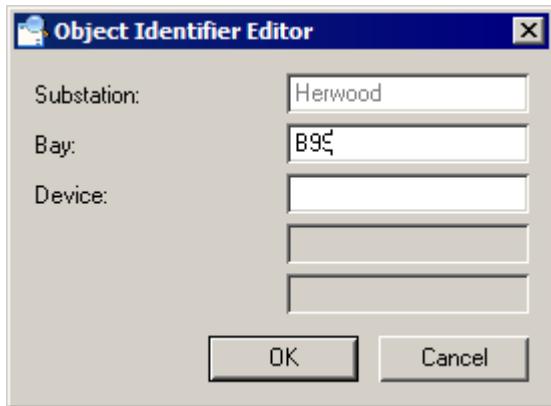


Figure 56: Object Identifier editor

16.3.14 Moving objects

To move an application object:

1. Choose the application from where to move the object type and the object. Several objects can also be moved by selecting them at the same time with the help of the CTRL key.
2. Choose **Cut** from the **Edit** menu.
3. Choose another application where the object or the objects should be inserted.
4. Choose **Paste** from the **Edit** menu.



It is possible to cut a maximum of 1000 objects at one time.

16.3.15 Deleting application objects



If only a process object group and no indices is selected, the whole group and ALL INDICES under it will be deleted.



The Delete operation cannot be undone!
A maximum of 10 000 objects can be deleted at a time.

To delete an application object:

1. Choose the application from which objects should be deleted.
2. Choose the object type and the objects to be deleted. Several objects can be selected by holding the CTRL key down while clicking the objects or pressing the mouse button down and dragging the pointer over the objects. In the case of a process object, choose the object group and then the indexes to be deleted. Note that if only a process object group and no indexes is selected, the whole group and all indexes under it will be deleted.
3. Choose **Delete** from the **Edit** menu. The delete prompt appears.
4. Click **OK** to delete or **Cancel** to cancel the deletion.

If copied application objects are going to be deleted, the dialog box shown in [Figure 57](#) appears. In the dialog box, the user has to confirm whether the delete operation is continued or not. In this case, continuing the operation will cause the copied application object to disappear also from the Clipboard.

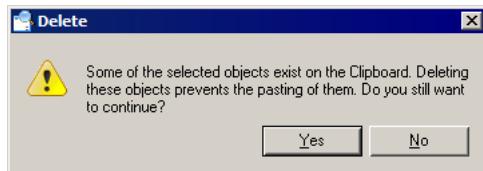


Figure 57: The dialog box, that appears when a copied application object is being deleted.

16.3.16 Installing standard functions

To enter the Standard Function Installation Tool:

1. Click **Install Standard Function...** in the Object menu.
2. The Standard Function Installation Tool dialog opens. For further instructions, see the SYS600 Picture Editing manual.

16.3.17 Documenting application objects

To enter the **Document Tool**:

1. Click **Document...** in the menu.
2. The **Document Tool** dialog opens. For further instructions, see the SYS600 Installation and Administration manual.

16.3.18 Exporting and importing application objects

To enter the **Export and Import Tool**:

1. Click **Export...** or **Import...** in the menu.
2. The **Export and Import Tool** dialog opens. For further instructions, see the SYS600 Installation and Administration manual.

16.3.19 Searching objects and files

To enter the **Search Tool**:

1. Click **Search...** in the menu.
2. The **Search Tool** dialog opens. For further instructions, see the SYS600 Installation and Administration manual.

16.3.20 File transfer

File transfer is implemented via process database. It is possible to create File Transfer LAG 1.4 Process Objects in the same way as other Process Objects.

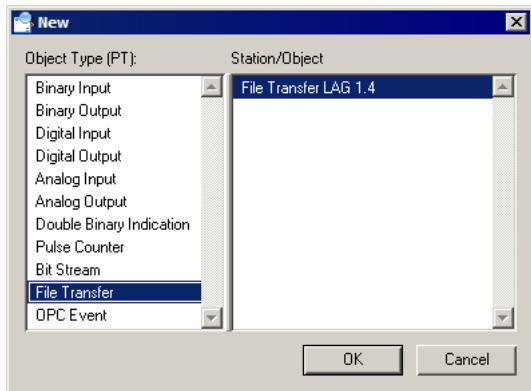


Figure 58: File transfer objects are created in the same way as other Process Objects

It is not possible to set the address for File Transfer LAG 1.4, only the station UN can be modified. There is a separate page for File Transfer objects under the **Dynamic** tab. See [Figure 59](#).

The following functions are supported in current release:

- Receiving (uploading) a file from a station
- Sending (downloading) a file to a station
- Browsing the file hierarchy of a station
- Reading file attributes from a station
- Deleting a file or directory in a station

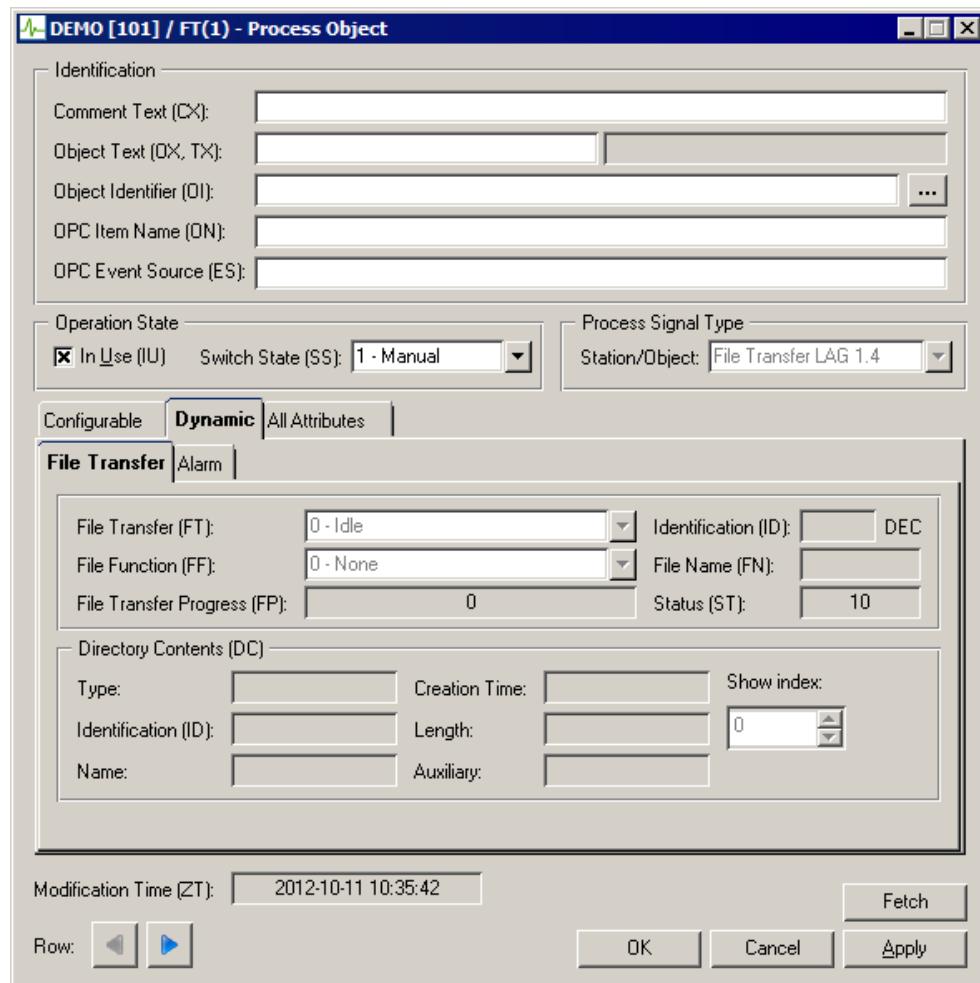


Figure 59: There is an own page for File Transfer objects under the Dynamic tab

The process objects of this type have all the same common attributes as other process objects. Especially, the post-processing attributes, such as EE, AE etc., may be used to report the completion of the transfer to the application.

The address attributes OA and OB have no meaning in conjunction with FT objects. OA should be set to zero (if set at all).

There may be any number of FT objects connected to one station, for example, each configured to a specific download/upload.

16.4 General principles for using object definition tools

16.4.1 Definition tool title

The title of an object definition tool shows the name of the object and the object type. Furthermore, it shows the signal type for Process Objects, the event handling type for Event handling Objects, the scaling algorithm for Scale Objects, the logging function for Data Objects, the primary object for Event Channels and the logging profile type for Logging Profile Objects. It also shows the name and number of the application where from the tool is started. In the case of process objects, the object name is followed by the index.

16.4.2 Pages

Most of the object definition tools are composed of pages, which can be accessed by clicking the tabs. In some cases, the tool contains so many pages that not all of them can be shown on screen at the same time. In these cases, make the tabs visible by moving to the left and right with the arrow keys.

[Figure 60](#) shows an example of an object definition tool, the data object definition tool. It has five visible tabs.

16.4.3 User interaction

The user interface of the object definition tools follows Windows standard interface.

Options that cannot be applied to the object in question, for example because of some previous selection, are made unavailable. In [Figure 60](#), the Source and Pulse Scale texts are disabled, which indicates that these features cannot be changed.

Text boxes that provide a drop-down list of options are equipped with a down arrow. Make a choice by clicking the arrow and then selecting an option on the list, or by clicking the arrow and holding the mouse button down while dragging the cursor to the desired option and releasing the button). In [Figure 60](#) the Logging function has a drop-down list of options. A button with three dots opens a selector dialog where a choice can be made by clicking in a selection list. In [Figure 60](#), the button is besides the Source text box.

Check boxes are used when an attribute or feature can be either active or inactive. A cross in the box means that the attribute or feature is selected. Change the selection by clicking the check box.

The attributes are described in brief in the tool descriptions in the next sections. For a detailed description of the attributes, refer to the corresponding descriptions in Sections 4 .. 12.

16.4.4 Storing the object and exiting the tool

When object definition is complete, the user can save the object and exit the tool, cancel the definitions and exit the tool, or save the definition without exiting the tool. These options are at the bottom of the dialog:

OK	Saves the object definition (updates the object if it was changed) and closes the tool.
Cancel	Cancels the definition and closes the tool.
Apply	Saves the definition but does not close the tool.

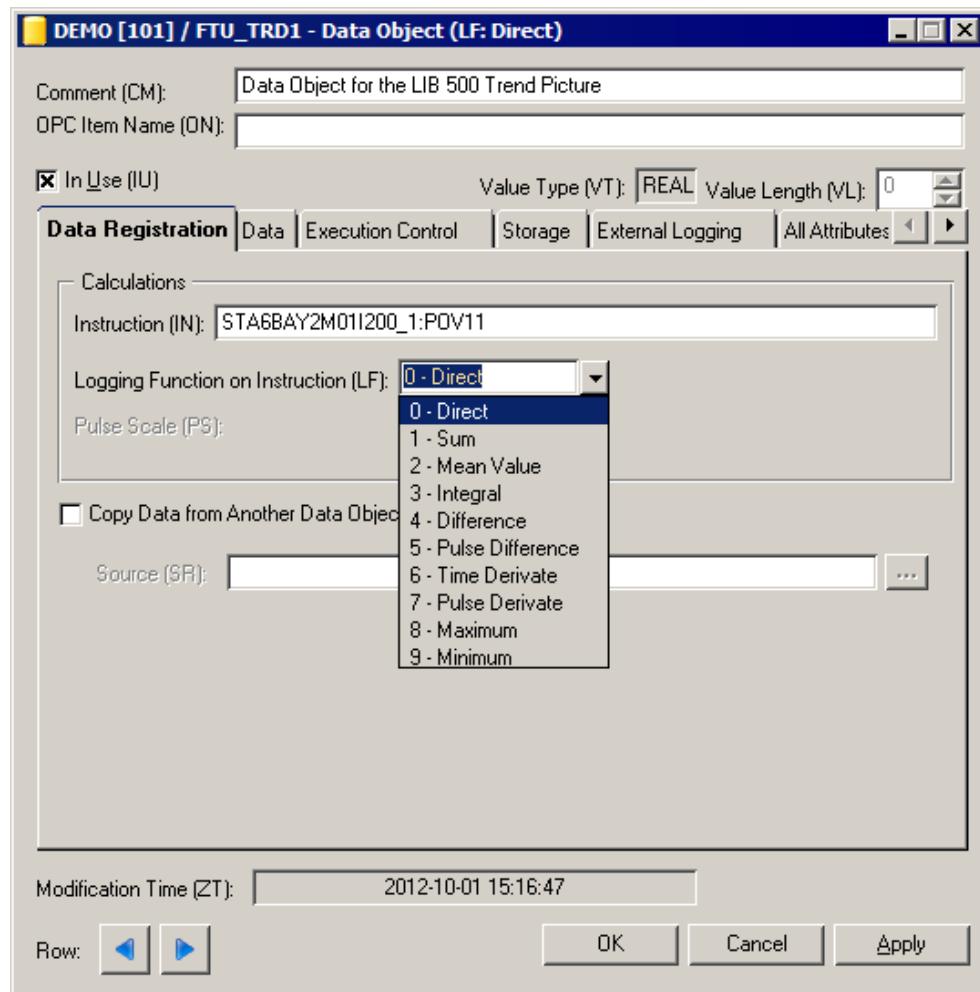


Figure 60: An example page of a definition tool

16.4.5 Connecting and disconnecting objects

An application object can be connected to another application object by an object definition tool. For example, a process object index can be connected to an event handling object, a scale or an event channel by the process object definition tool, a data object or command procedure can be connected to a time channel by the corresponding tool. On the other hand, SYS600 base system allows several process object indexes to be connected to one event handling object, scale and event channel, and several data objects and command procedures to be connected to one time channel. To ease engineering work, a way to connect several objects to a single object in one go has been implemented in Object Navigator.

As an example, process object indexes 1, 2 and 3 in EXAMPLE group in DEMO application will be connected to EXAMPLE event channel:

1. Open **Event Channel** tool.
2. Click the **Connected objects** page. If the page is empty, only **Add...** button is enabled.
3. Click the **Add...** button. Empty **Selected Objects** page appears.
4. Click **Select...** button. **Object Selector** that shows process objects in DEMO application will be opened. Select indexes 1, 2 and 3 in EXAMPLE group, see [Figure 61](#). The process object definition tool is also available in **Object Selector**. However, the tool is in read-only mode, because the **OK** and **Apply** buttons are dimmed.
5. Click the **OK** button in **Object Selector**. **Object Selector** will be closed and **Event Channel** tool shows selected indexes as highlighted on Selected Objects page, see [Figure 62](#). To

- cancel the selection, click the **Cancel** button in the **Object Selector** or the **Event Channel** tool.
6. Click the **Connect** button. Information dialog appears. The connect operation is immediate, if the **Yes** button is clicked. It means that the AN attribute of the process object indexes will be set to EXAMPLE immediately, not after the **OK** or **Apply** button in the tool is clicked. The dialog also shows that the connect operation can be undone. Information dialog appears only if more than one object will be connected, see [Figure 63](#).
 7. Click the **Yes** button. The information dialog will be closed and **Connected objects** page pops up showing connected indexes. In addition to **Add...** button, **Undo** button is enabled, see [Figure 64](#).
 8. To cancel the operation, click the **Undo** button and select **Undo Connect** from pop-up menu.

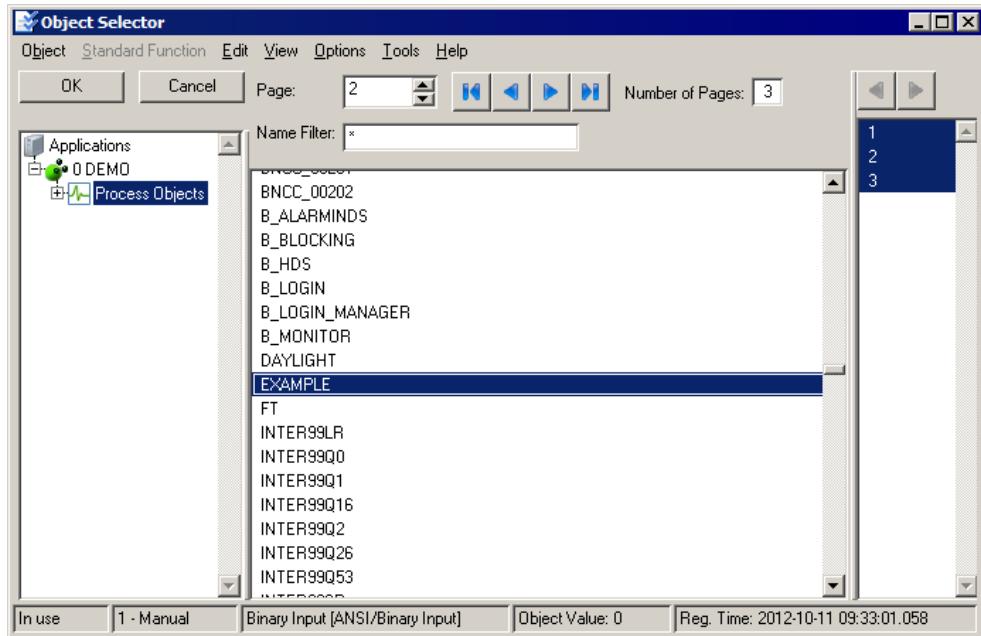


Figure 61: Object Selector

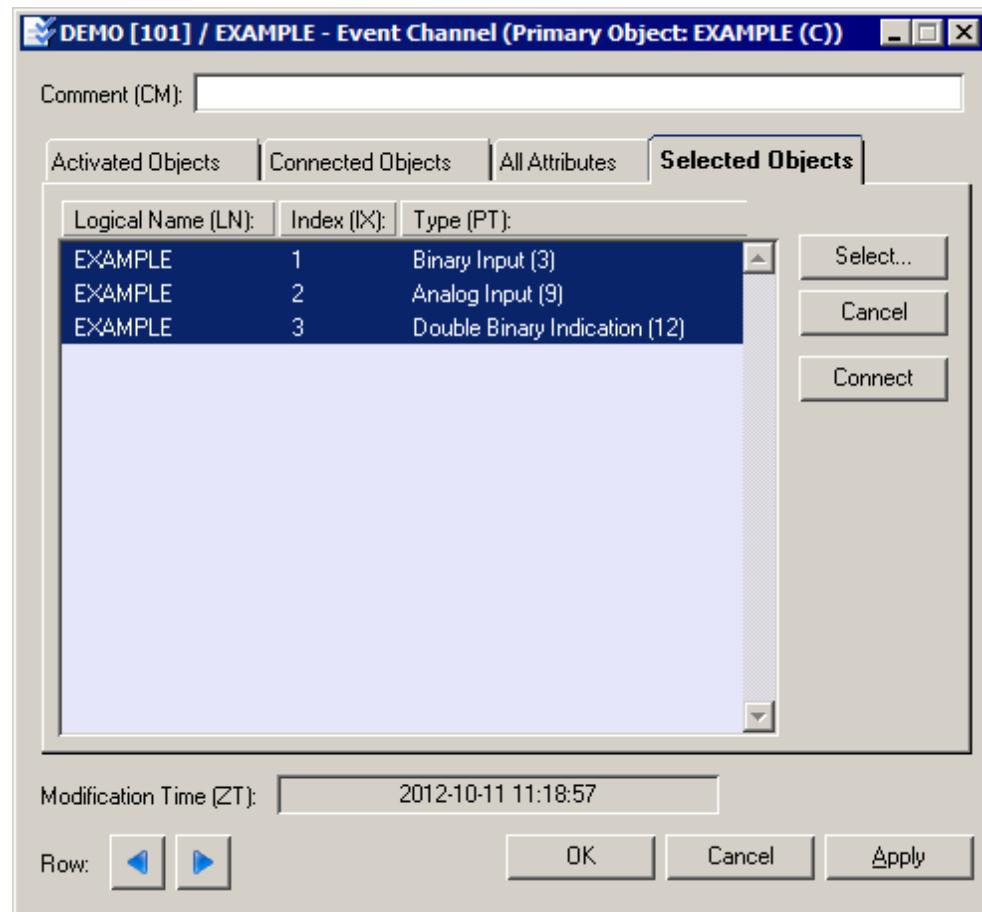


Figure 62: Selected Objects page

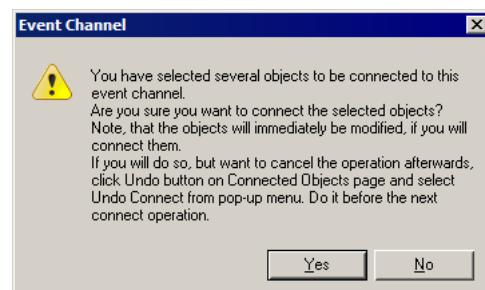


Figure 63: Info dialog for objects to be connected to an event channel.

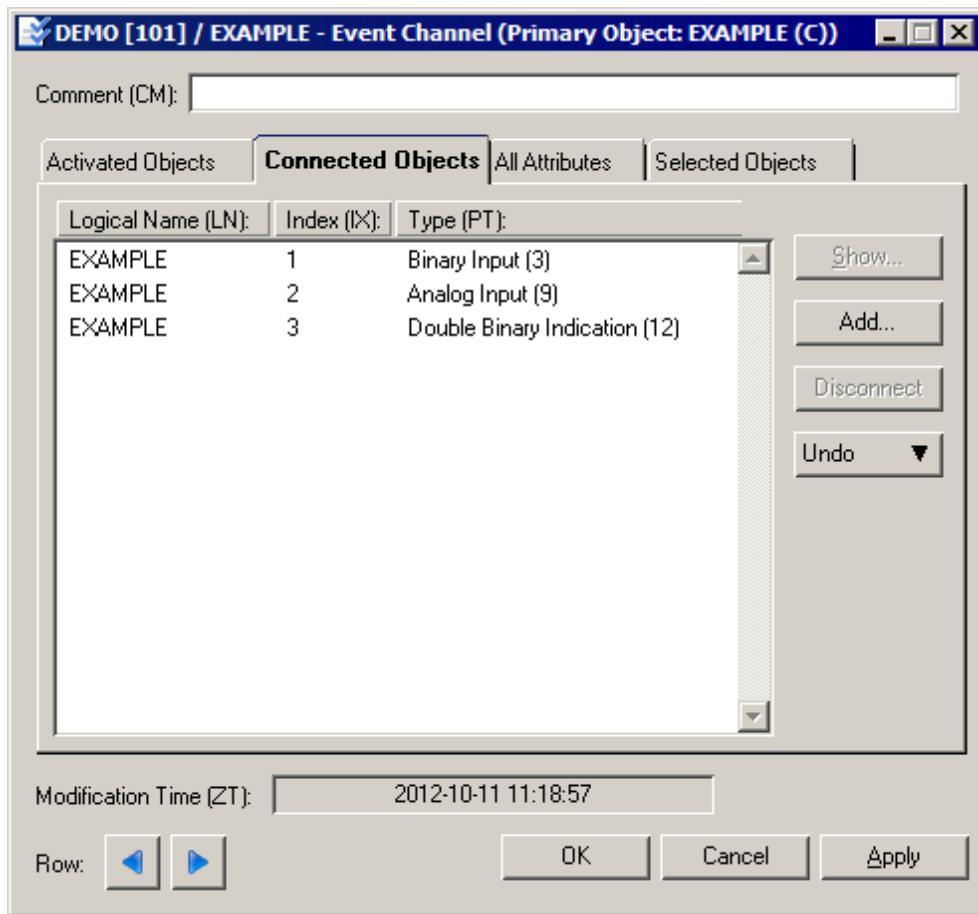


Figure 64: Connected Objects page after a connect operation.

Process object indexes 1 and 3 in EXAMPLE group in DEMO application will be disconnected from EXAMPLE event channel the following way:

1. Open the **Event Channel** tool.
2. Click the **Connected objects** page. Select indexes to be disconnected (see [Figure 65](#)).
3. Click the **Disconnect** button. Information dialog appears. The disconnect operation is immediate, if **Yes** button is clicked, which means that the AN attribute of the process object indexes will be set to empty immediately, not after **OK** or **Apply** button in the tool is clicked. The dialog also shows that the disconnect operation can be undone. Information dialog appears only if more than one object will be disconnected.
4. Click the **Yes** button. The information dialog will be closed and the **Connected objects** page shows the still connected indexes, if any, and **Undo** button is enabled.
5. To cancel the operation, click the **Undo** button and select **Undo Disconnect** from the pop-up menu.

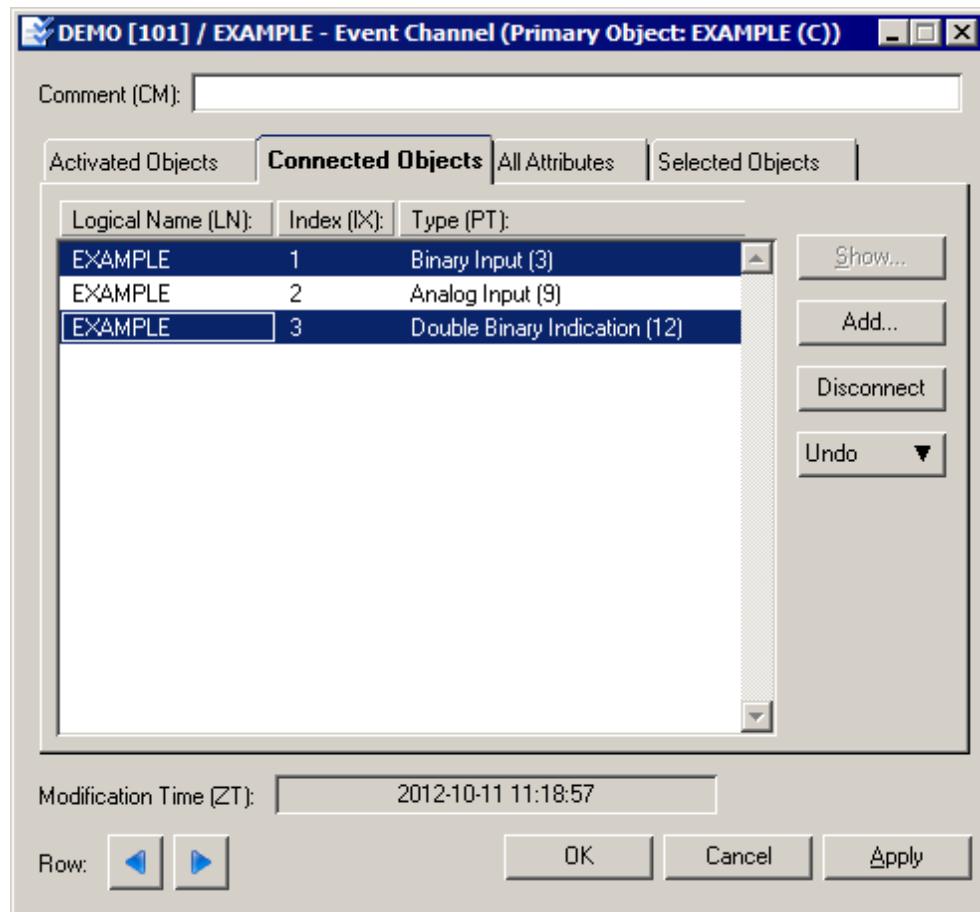


Figure 65: Two indexes on Connected Objects page have been selected to be disconnected from EXAMPLE event channel.



Connect and disconnect operations should be applied carefully, because they are very powerful. Especially, if objects are already connected to another object, a careless reconnect operation may cause objects to work unexpectedly.

16.5 Application self-diagnostics

When SYS600 base system encounters problems, it displays appropriate messages in Notification Window and writes them in the log file SYS_ERROR.LOG. Self-diagnostics for application objects has been implemented to ease to interpret messages and to provide actions to solve problems. The **Self diagnostics** dialog opens when the user clicks a node of a conflicting object type in the tree of the application. At the moment, some diagnostics for load time conflicts of process database have been introduced. [Figure 66](#) shows object type conflicts.

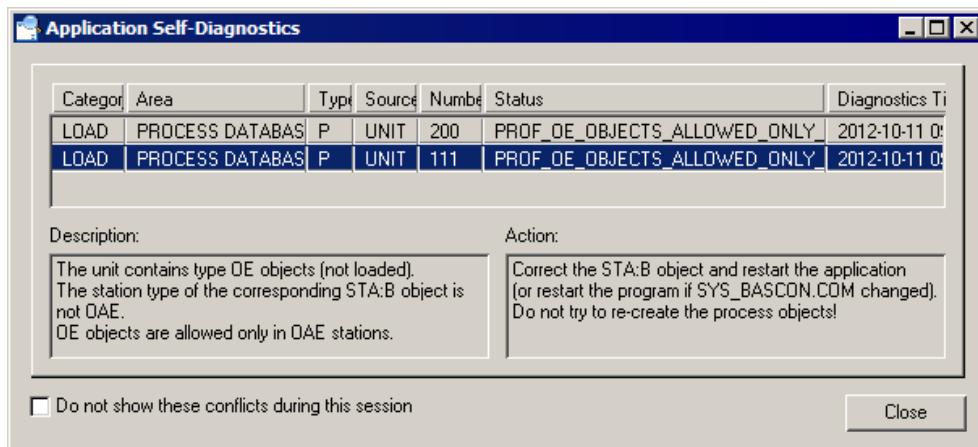


Figure 66: An example of self-diagnostics. The description field describes the problem on the highlighted line and the action field gives advises to solve the problem.

Section 17 Process Object Definition Tool

17.1 About this section

This section describes the definition tool for defining process objects of predefined types. It is divided into six sections as follows:

- | | |
|------------------------------|--|
| Section 17.2 | General : a summary of the definition tool. |
| Section 17.3 | Common area : the attributes and options in the common area of the tool. |
| Section 17.4 | Configurable attributes : the Configurable page and its sub-pages. |
| Section 17.5 | Dynamic attributes : the Dynamic page and its sub-pages. |
| Section 17.6 | All attributes : the All attributes page. |
| Section 17.7 | Object Identifier editor |

For a detailed description of the attributes mentioned in this section, refer to [Section 5](#).

17.2 General

The tool for defining process objects of predefined types is accessed from the **Object Navigator** by double-clicking a process object. Process objects are created in the **Object Navigator** (for the procedure, see [Section 16.3](#)). The **Process Object Definition Tool** can also be accessed from the Event Channel Definition Tool and the Event Handling Definition Tool.

The **Process Object Definition Tool** contains a common area and three main tabs:

- **Configurable**. Below this tab there are a number of sub-tabs each of which represent a page with configurable attributes.
- **Dynamic**. This tab presents several pages that show the values of the dynamic attributes at the moment the tool is opened. The content of the tool can also be updated.
- **All attributes**. This page lists all attributes of a process object type in alphabetical order.

The page **All attributes** has no sub-pages. The other two pages, **Dynamic Attributes** and **Configurable Attributes**, have sub-pages, which vary depending on the process signal type. The names of the sub-pages, as well as their contents, differ.

17.3 Common area

17.3.1 General

The common area above and below the pages, see [Figure 67](#), contains basic definitions that are common to all object types.

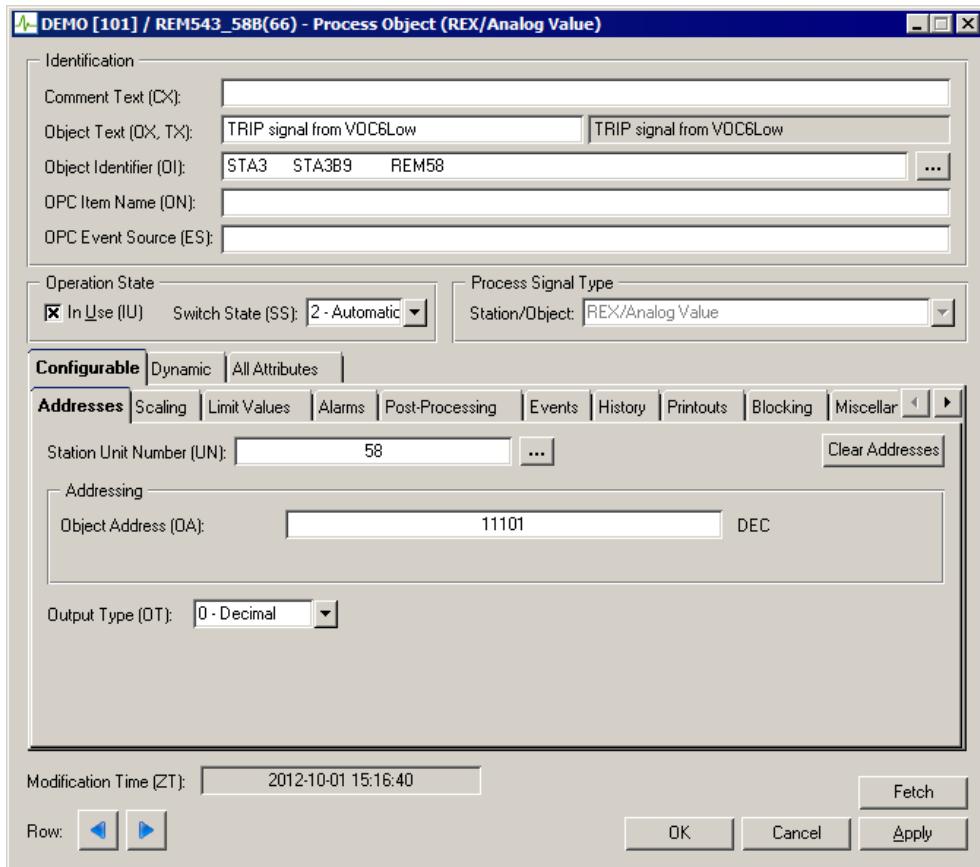


Figure 67: The Process Object Definition Tool. This picture shows the part of the tool that is common for all process objects. It also shows notebook pages, which differ according to the process signal type.

17.3.2 Identification

This section of the tool specifies the identification attributes of the objects, see [Section 5.2.2](#).

The **Object Identifier (OI)** attribute value contains the structural information, that is, certain character positions of OI value contain, for example, Substation, Bay or Device identification. To handle this structural nature of the attribute a dedicated Object Identifier editor is introduced, see [Section 17.7](#). The editor may be started by clicking the button with three dots on the right side of the Object Identifier text box, see [Figure 67](#).

17.3.3 Operation state

Take the object into use and out of use with the **In Use** check box. The selection is applied to the process object when **OK** or **Apply** is clicked. The selection specifies the IU attribute. The Switch State is the SS attribute (see [Section 5.2.4](#)).

17.3.4 Process signal type



If the object is not in use, changes to some dynamic attributes will not be updated.

This section presents information about the type of station, which reads or controls the object, and the object type as it is defined in the station. This is not an attribute.

The structure of the tool depends on the signal type, as mentioned earlier. However, process objects that have not been created by **Object Navigator** or an application library (LIB 5xx or Power Process Library) may have no knowledge about the signal type. It is possible in this kind of situation to fix the signal type by selecting a type from the drop-down list (see [Figure 68](#)). The tool structure will immediately be changed. When the selection has been confirmed by clicking the **Apply** or **OK** button, the signal type is permanent and cannot be changed by the tool.

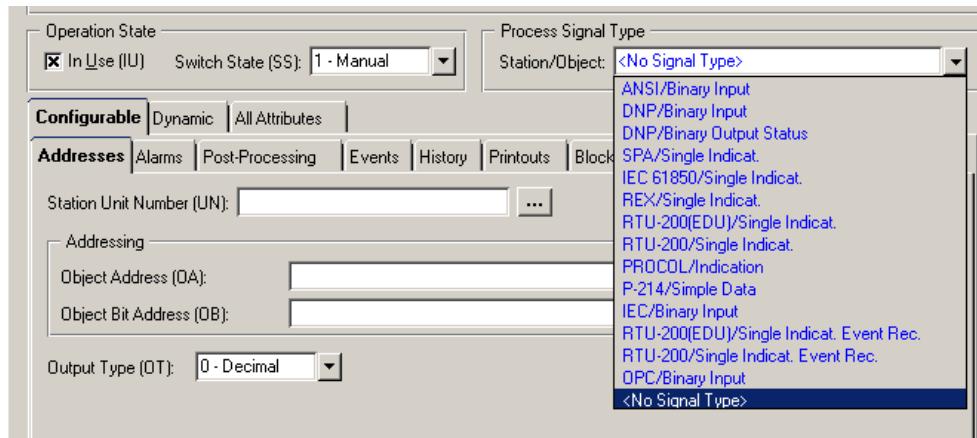


Figure 68: Selection of a signal type.

17.3.5 Other information

The **Modification Time** text box below the pages shows the time when the object was last modified, regardless of whether it was modified in the tool or with SCIL commands (CREATE, MODIFY).

The **Next** and **Previous Process Object Index (IX)** and **Group (LN)** command buttons below the pages make it easier to move from one Process Object Index or Group to another. When any of the four command buttons is active, there is one or more object(s) or index(es) in the respective direction of the button.

Values shown in the tool are the values at the moment the tool was opened. To update the values of the tool, click the **Fetch** button.

17.4 Configurable attributes

17.4.1 Overview

The configurable attributes are grouped into the following sub-pages, which may be shown or hidden depending on the object type:

- **Addresses**
- **Scaling** (analog objects and pulse counters)
- **Limit values** (analog objects)
- **Alarms**
- **Events**
- **History**
- **Printouts**

- **Blocking**
- **Topology**
- **Miscellaneous**

17.4.2 Addresses

In the **Addresses** page (see [Figure 69](#)) the addressing attributes are defined (see [Section 5.2.3](#)). The appearance of the page varies according to the object type. The appearance also depends on the encoding of the addresses of the unit (attributes OA and OB).

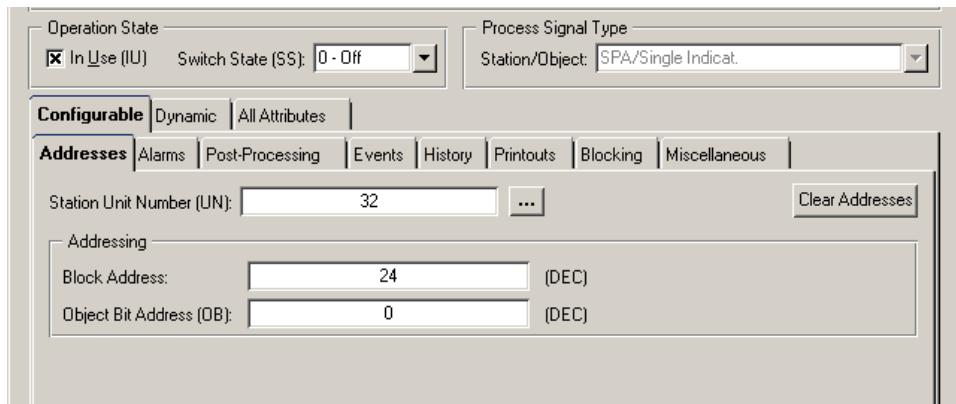


Figure 69: Example of the attributes that are defined in the Addresses page.

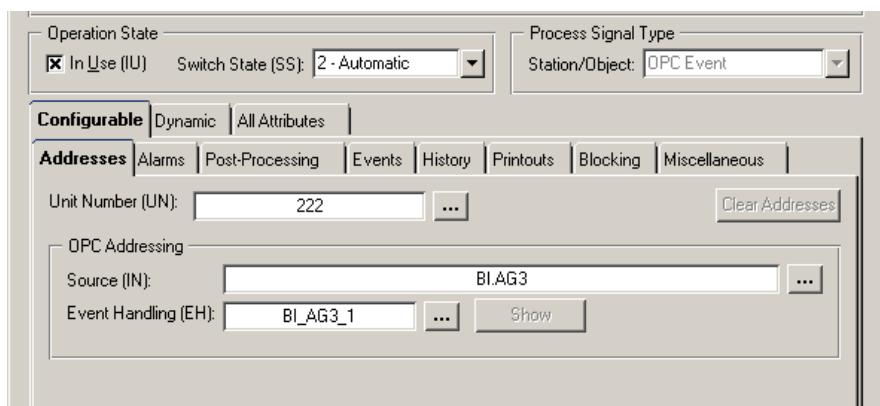


Figure 70: Example of the attributes that are defined in the Addresses page of OPC Event object.

17.4.3 Scaling

The **Scaling** page is shown only for analog objects and pulse counters (see [Figure 71](#)). See [Section 5.2.5](#) for the attributes.

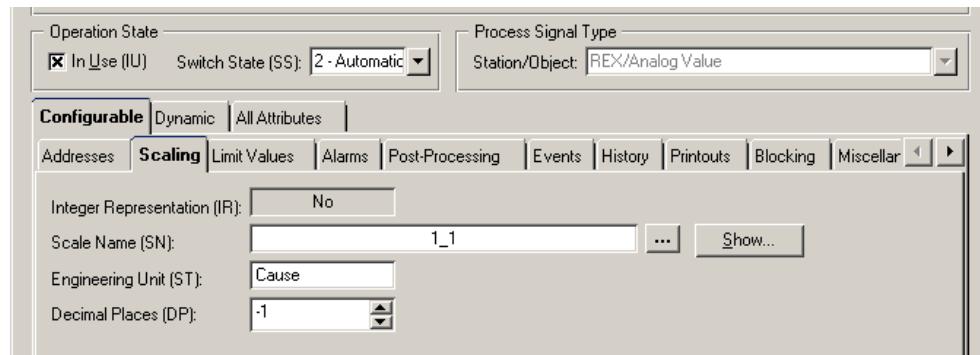


Figure 71: The Scaling page.

17.4.4 Limit values

The **Limit values** page (see Figure 72) for limiting value supervision is present when defining an analog object. See [Section 5.2.7](#) for the attributes. The pages for analog input objects are different from the pages for analog output objects.

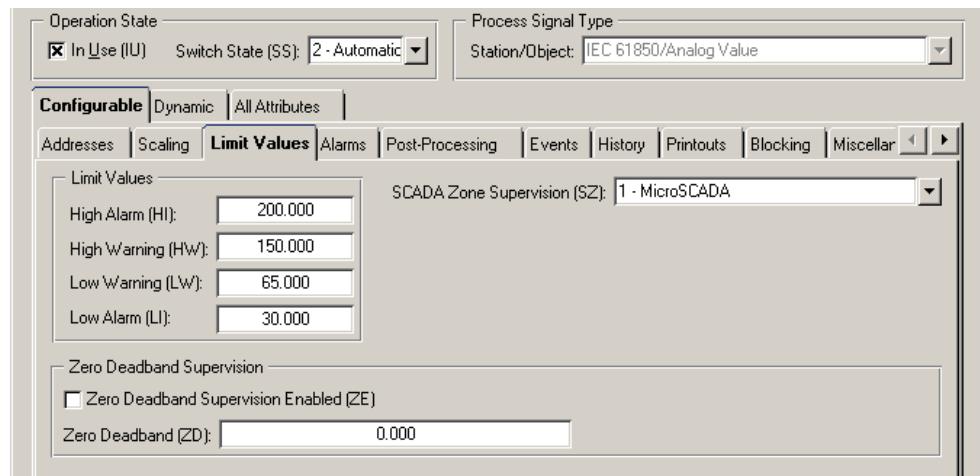


Figure 72: The Limit Values page for a process object of type analog input.

17.4.5 Alarms

There are three different **Alarms** pages for defining alarms:

- Binary input page: See [Figure 73](#)
- Double binary indication page: See [Figure 74](#)
- Page for other objects: See [Figure 75](#)

See [Section 5.2.6](#) for the attributes.

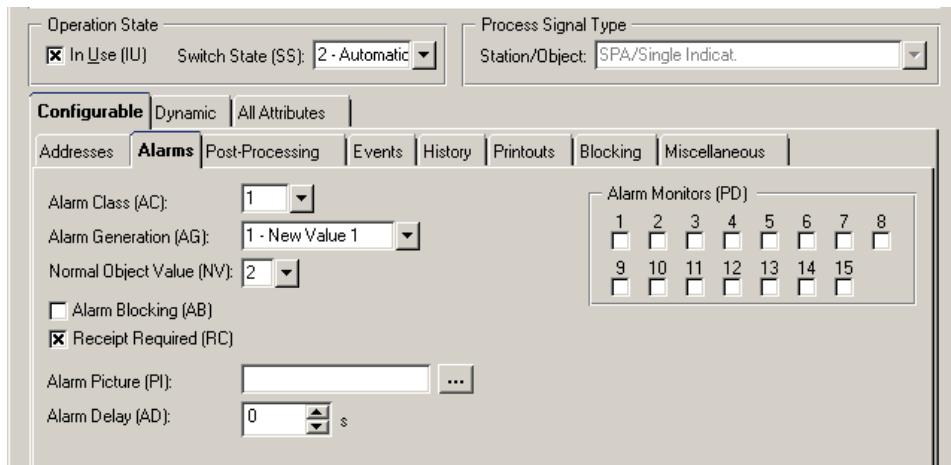


Figure 73: The Alarms page of a binary input object.

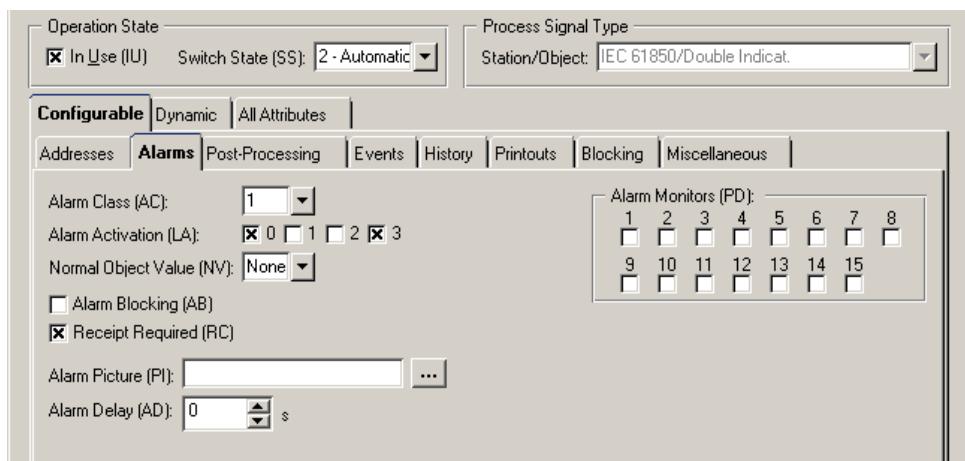


Figure 74: The Alarms page of a double binary indication object.

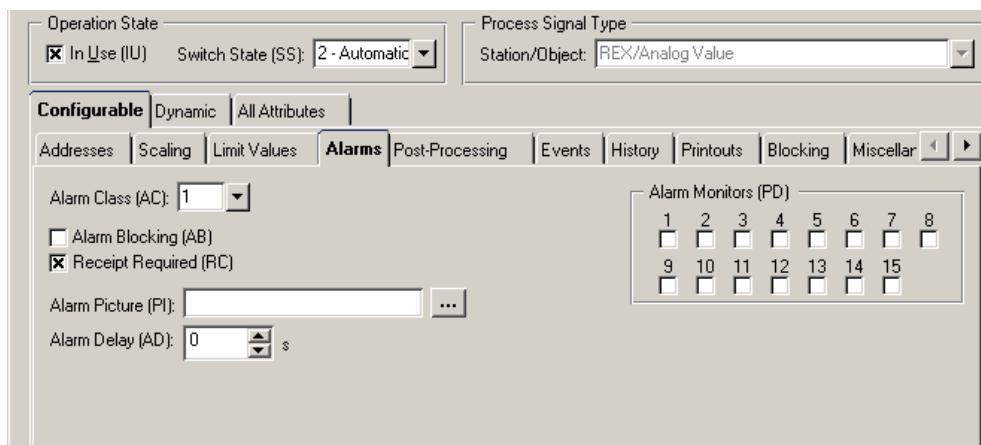


Figure 75: The Alarms page of objects, which are not of type binary input or double binary indication.

17.4.6 Post-processing

The **Post-processing** page (see [Figure 76](#)) is shown for all object types. See [Section 5.2.8](#) for the attributes.

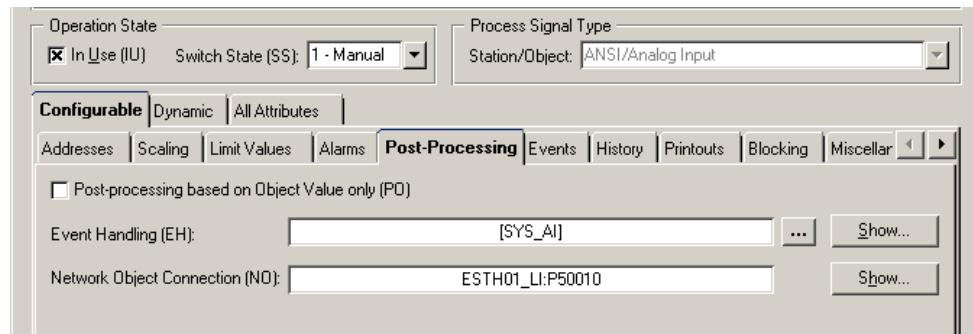


Figure 76: The Post-Processing page.

17.4.7 Events

The **Events** page (see [Figure 77](#)) is shown for all object types. See [Section 5.2.9](#) for the attributes.

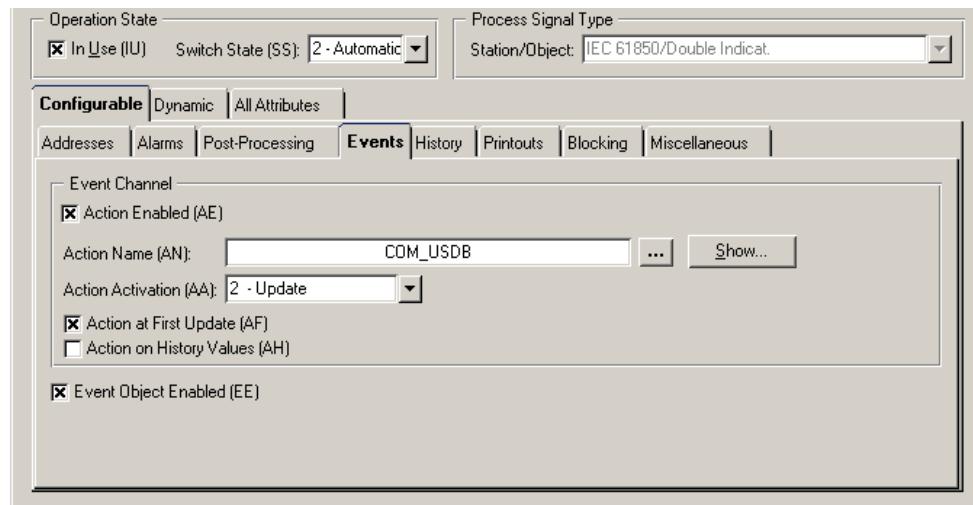


Figure 77: The Events page.

17.4.8 History

The **History** page (see [Figure 78](#)) is shown for all object types. See [Section 5.2.10](#) and [Section 5.2.11](#) for the attributes.

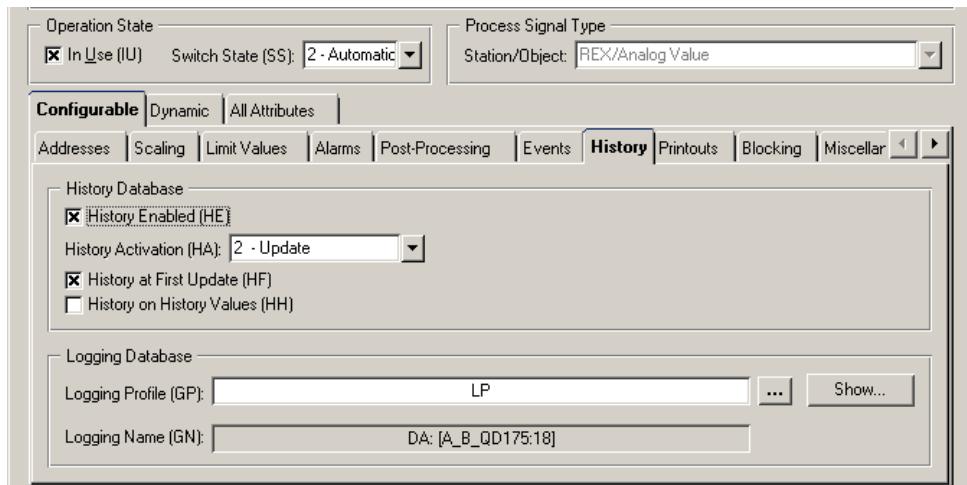


Figure 78: The History page.

17.4.9 Printouts

The **Printouts** page (see [Figure 79](#)) is shown for all object types. See [Section 5.2.13](#) for the attributes.

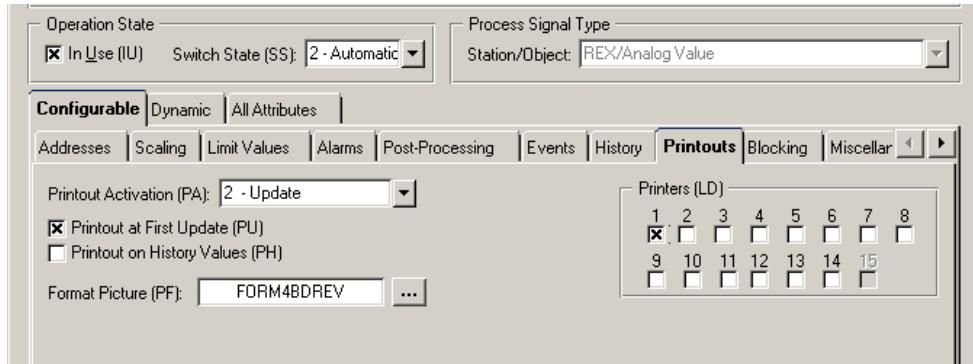


Figure 79: The Printouts page.

17.4.10 Blocking

The **Blocking** page (see [Figure 80](#)) is shown for all objects type. See [Section 5.3.4](#) for the attributes.

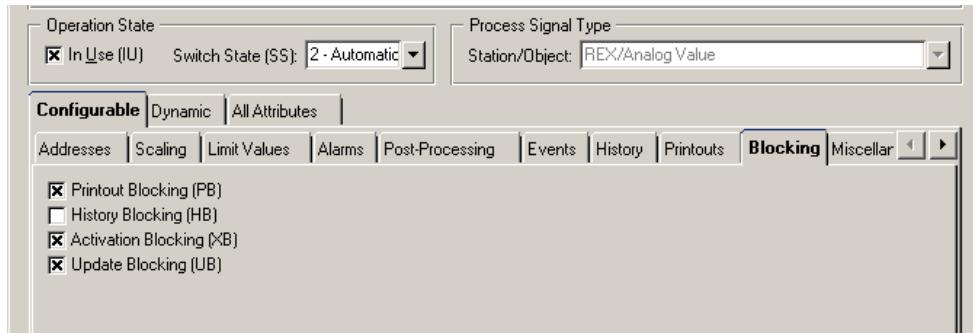


Figure 80: The Blocking page.

17.4.11 Topology

The **Topology** page (see [Figure 81](#)) shows attributes of topology objects. See [Section 5.2.14](#) for attributes.

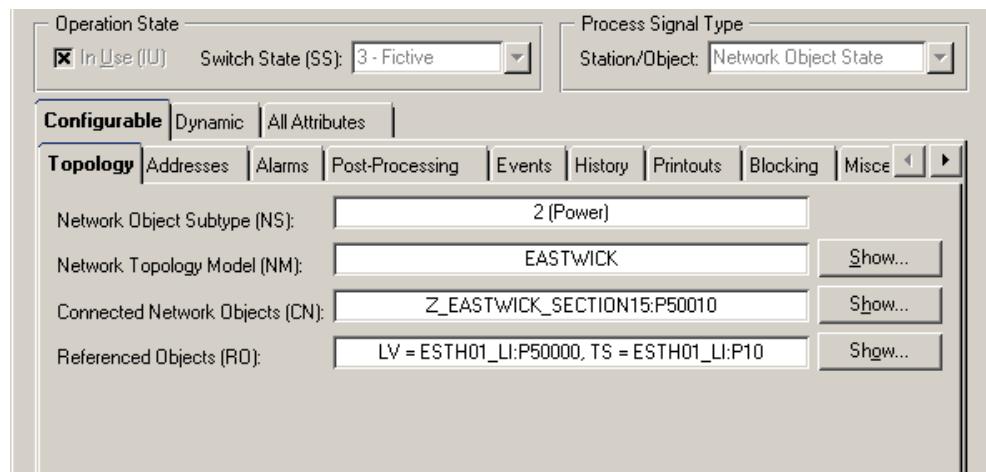


Figure 81: The Topology Page.

17.4.12 Miscellaneous

The **Miscellaneous** page (see [Figure 82](#)) is shown for all object types. See [Section 5.2.15](#) for the attributes.

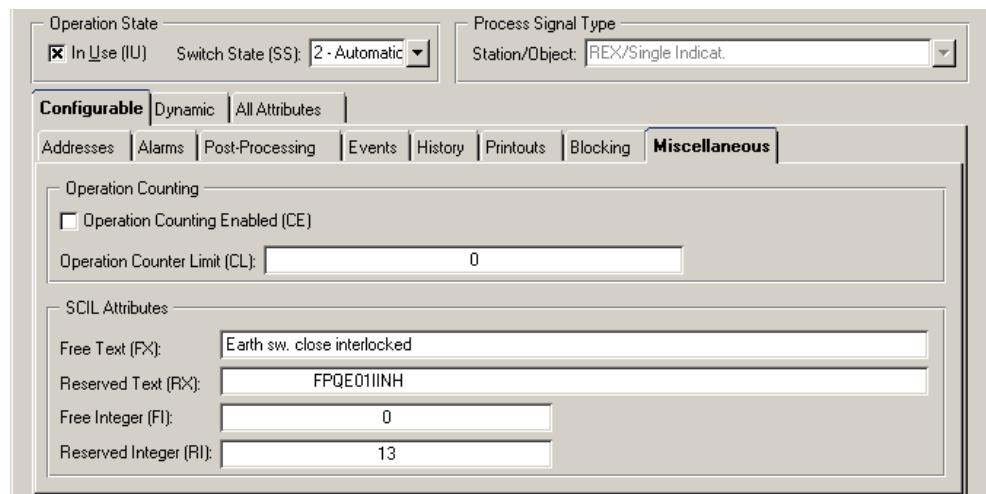


Figure 82: The Miscellaneous page.

17.5 Dynamic attributes

17.5.1 Object state

The **Object State** page is shown for all object types except for file transfer objects. See [Section 5.3.1](#) for the attributes.

There are four different Object State pages:

- Binary input and double binary indication page: See [Figure 83](#)
- Page for output objects: See [Figure 84](#)
- Topology page: See [Figure 85](#)
- Page for other objects: See [Figure 86](#)

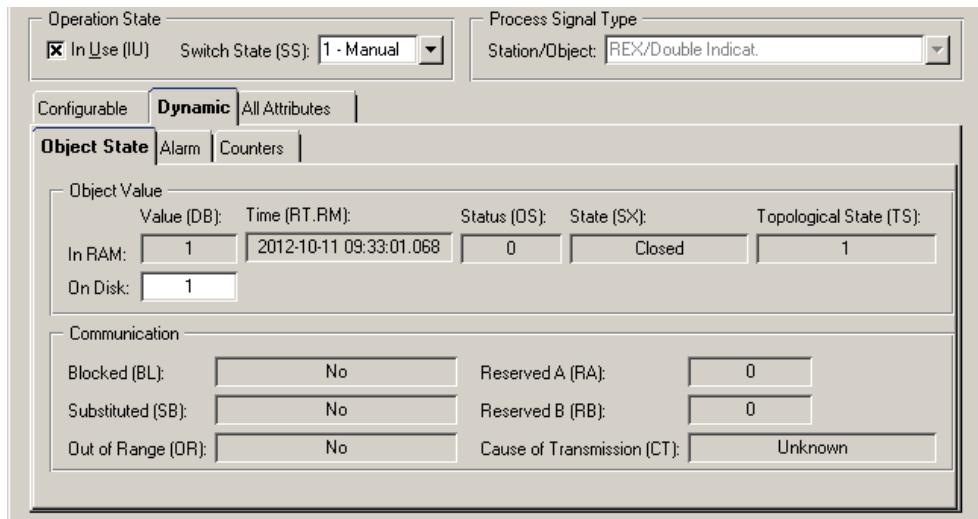


Figure 83: The Object State page of Double Binary Indication.

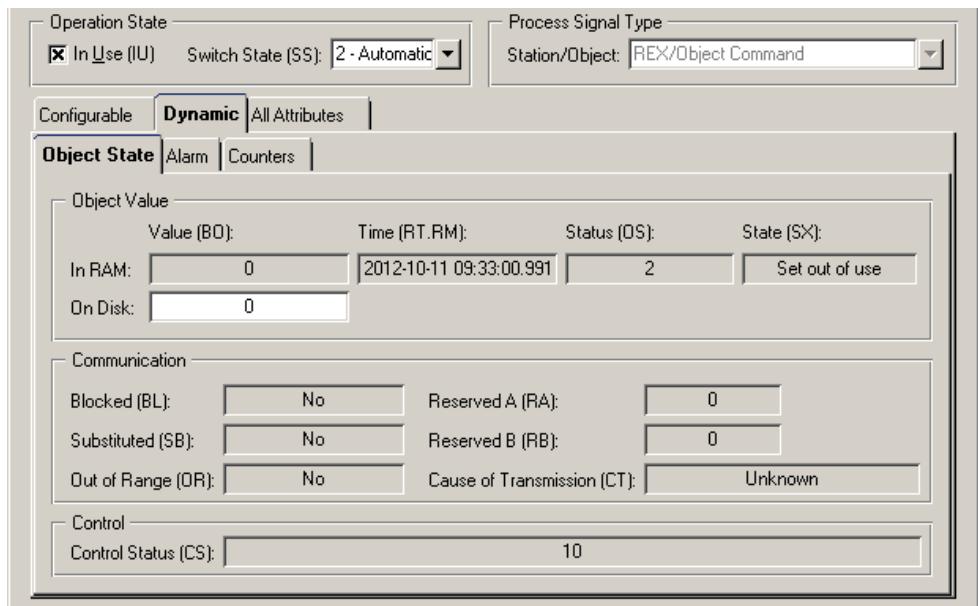


Figure 84: The Object State page of Binary Output.

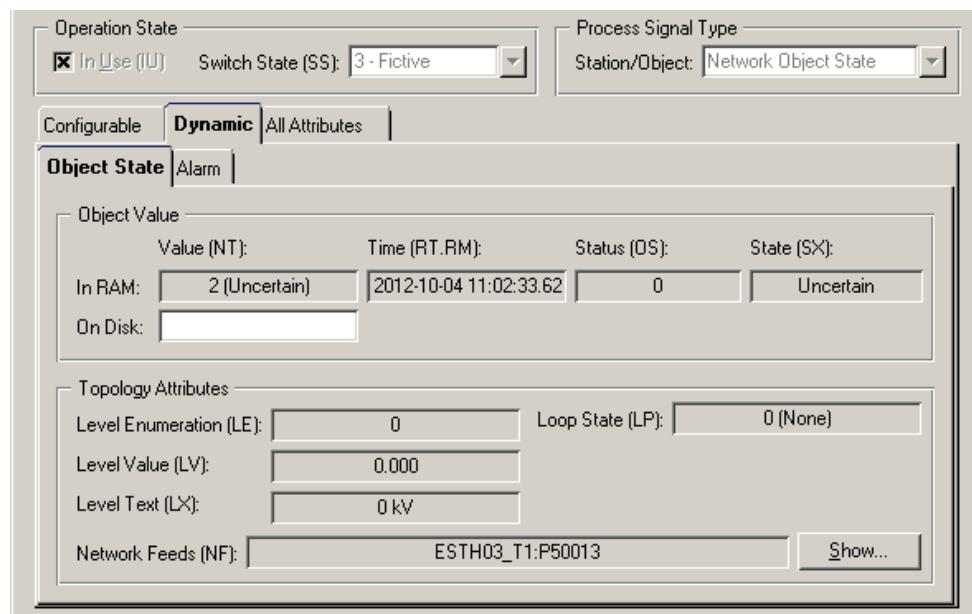


Figure 85: The Object State page of topology objects.

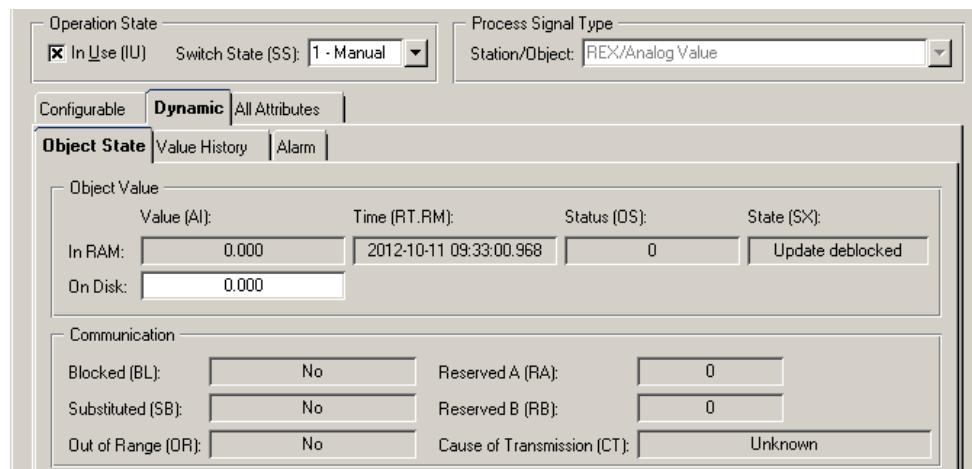


Figure 86: The Object State page of other objects.

17.5.2 File transfer

The **File Transfer** page is visible only for the file transfer objects. See [Section 5.3.11](#) for the attributes.

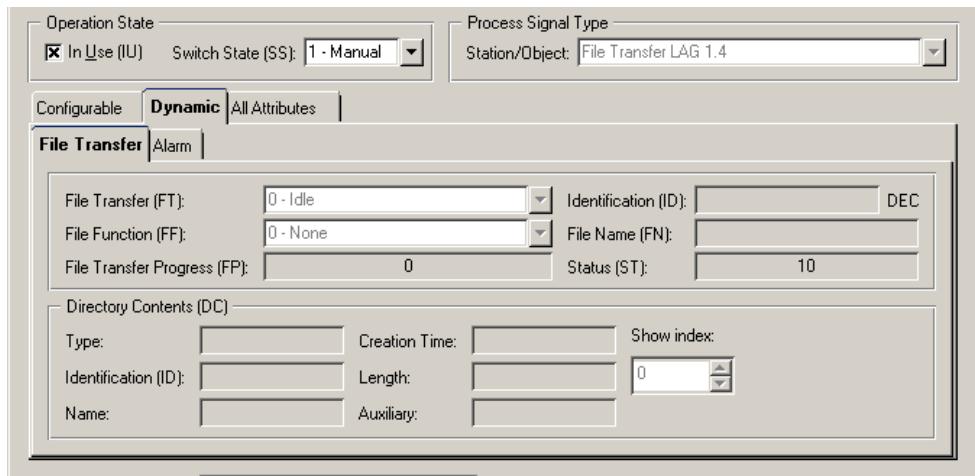


Figure 87: The File Transfer page.

17.5.3 Value history

The **Value History** attributes are analog input (AI) specific. See [Section 5.3.6](#) for the attributes.

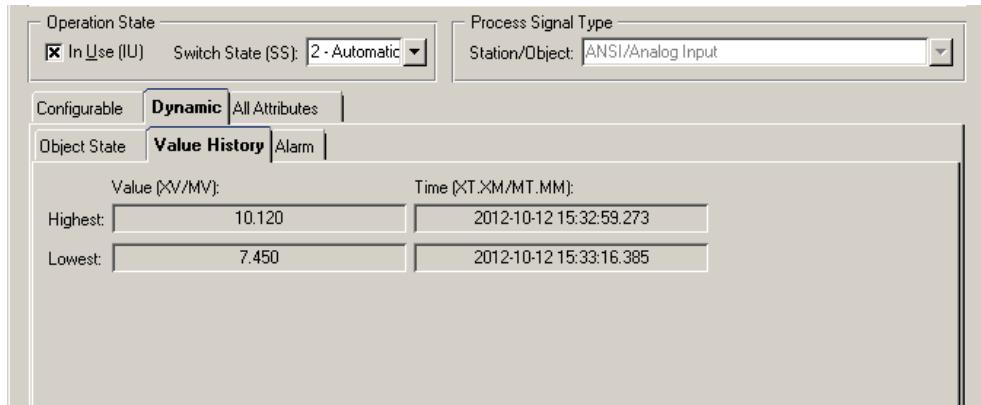


Figure 88: The Value History Page.

17.5.4 OPC event

The **OPC Event** page (see [Figure 89](#)) is shown for OPC Event (OE) objects. See [Section 5.3.10](#) for the attributes.

The screenshot shows the 'OPC Event' page of the Process Object Definition Tool. At the top, there are two dropdown menus: 'Operation State' set to 'In Use (IU)' and 'Switch State (SS): 2 - Automatic', and 'Process Signal Type' set to 'OPC Event'. Below these are tabs for 'Configurable', 'Dynamic' (which is selected), and 'All Attributes'. Under 'Object State', the tabs are 'OPC Event' (selected), 'Alarm', and 'Counters'. The main area contains sections for 'Common Attributes for all Event Types' and 'Attributes for Condition Event'. In the 'Common Attributes' section, 'OPC Event Message (VM)' is 'Changed from high alarm to low alarm', 'OPC Event Type (VT)' is 'Condition Event', 'OPC Severity (SE)' is '9', 'OPC Event Time (VQ)' is '2012-10-16 08:48:06.244', and 'OPC Event Category (VC)' is '-87'. In the 'Attributes for Condition Event' section, 'OPC Condition Name (CN)' is 'AI AZ4', 'OPC Subcondition Name (SN)' is 'AI AZ4', 'OPC Change Mask (CM)' is '0', 'OPC New State (NS)' is '3', 'OPC Quality (QU)' is '00C0 (OK)', 'OPC Ack Required (AK)' is 'Yes', 'OPC Active Time (OQ)' is '2012-10-16 08:48:06.244', and 'OPC Cookie (CK)' is '0'. There is also a section for 'Attribute for Tracking or Condition Event' with 'OPC Actor ID (ID)'.

Figure 89: The OPC Event page.

17.5.5 Alarm

The **Alarm** page (see [Figure 90](#)) shows the present alarm state of the object. See [Section 5.3.3](#) for the attributes.

The screenshot shows the 'Alarm' page of the Process Object Definition Tool. At the top, there are two dropdown menus: 'Operation State' set to 'In Use (IU)' and 'Switch State (SS): 1 - Manual', and 'Process Signal Type' set to 'ANSI/Analog Input'. Below these are tabs for 'Configurable', 'Dynamic' (selected), and 'All Attributes'. Under 'Object State', the tabs are 'Value History' and 'Alarm' (selected). The main area contains fields for 'Alarm (AL)', 'Alarm State (AS)', 'Alarm Receipt (AR)', 'Alarm Zone (AZ)', 'Alarm Activated at (YT.YM)', 'Alarm Activated/Deactivated at (AT.AM)', and 'Warning Qualified Time (WQ)'. The values are: 'Alarm (AL)': Yes, 'Alarm State (AS)': 1, 'Alarm Receipt (AR)': No, 'Alarm Zone (AZ)': High Alarm, 'Alarm Activated at (YT.YM)': 2012-10-11 09:33:00.850, 'Alarm Activated/Deactivated at (AT.AM)': 2012-10-11 09:33:00.850, and 'Warning Qualified Time (WQ)': 2012-10-11 09:33:00.850 DS.

Figure 90: The Alarm page.

17.5.6 Counters

The **Counters** page (see [Figure 91](#)) shows the values of the operational counters. See [Section 5.3.5](#) for the attributes.

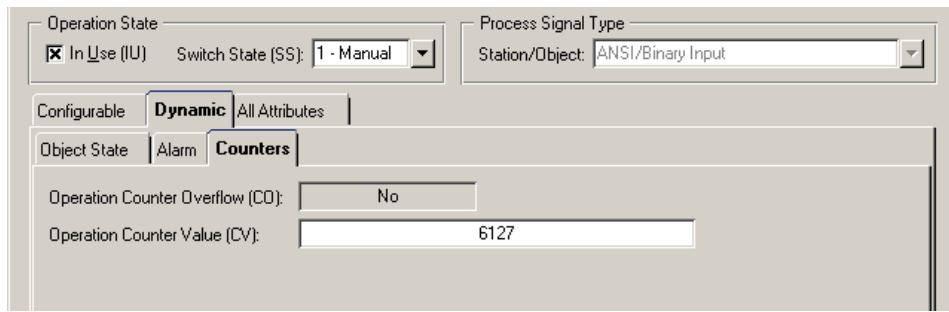


Figure 91: The Counters page.

17.6 All attributes

The **All Attributes** page (see [Figure 92](#)) lists all attributes of the process object type in question. The attribute values shown in a white text box can be edited. The attribute values shown on a gray background cannot be edited. When the attribute name is dimmed, the attribute can only have the default value and cannot be edited. For example, SN (Scale Name) attribute of Analog Input object when the value of AI attribute is represented as an integer (IR = 1).

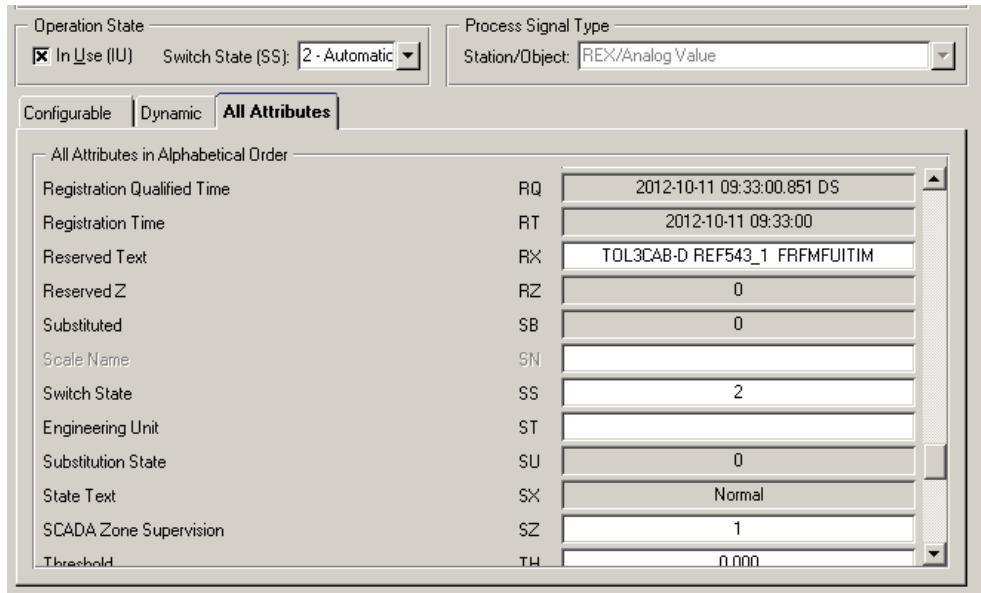


Figure 92: The All Attributes page.

17.7 Object Identifier editor

The **Object Identifier (OI)** attribute is divided into fields according to application specific conventions (See the System Objects manual for the application attribute OI). Due to this structural nature, a dedicated editor dialog is introduced, which knows the depth of the application's Object Identifier definition and the lengths of the fields. The [Figure 93](#) shows the OI attribute with three levels: **Substation**, **Bay** and **Device**. Up to five levels are supported.

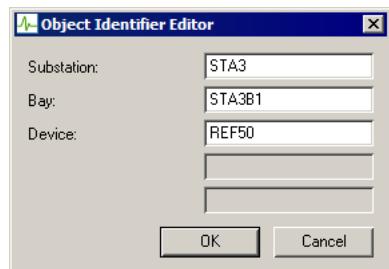


Figure 93: *OI Editor showing the OI attribute with three levels.*

Section 18 Event Handling Object Definition Tool

18.1 About this section

This section presents the definition tool for defining event handling, and provides a point-by-point description of the data and text boxes in the tool.

18.2 General

The **Object Navigator** has access to the tool for defining event handling by double-clicking an event handling object. A new object is created in the **Object Navigator**. The procedure is described in [Section 16.3](#).

The tool has the common area and five pages for the event handling of type SYS: **State Texts/Message Texts/OPC A&E Server/Connected objects** and **All Attributes**. The tool has six pages for the event handling of type AEC: The additional page is **Event Messages** for the object of SIMPLE event type and **Subcondition Names** for the object of CONDITION event type. The page **Connected objects** lists all process objects (if any) connected to the current event handling object. Selecting a row on the list and clicking the button or simply double clicking on a row can show the definition of a process object. The page **All Attributes** contains all attributes of the event handling object in alphabetical order.

18.3 Common area

The common area above and below the pages is shown in [Figure 94](#). See [Section 6.3](#) for the attributes.

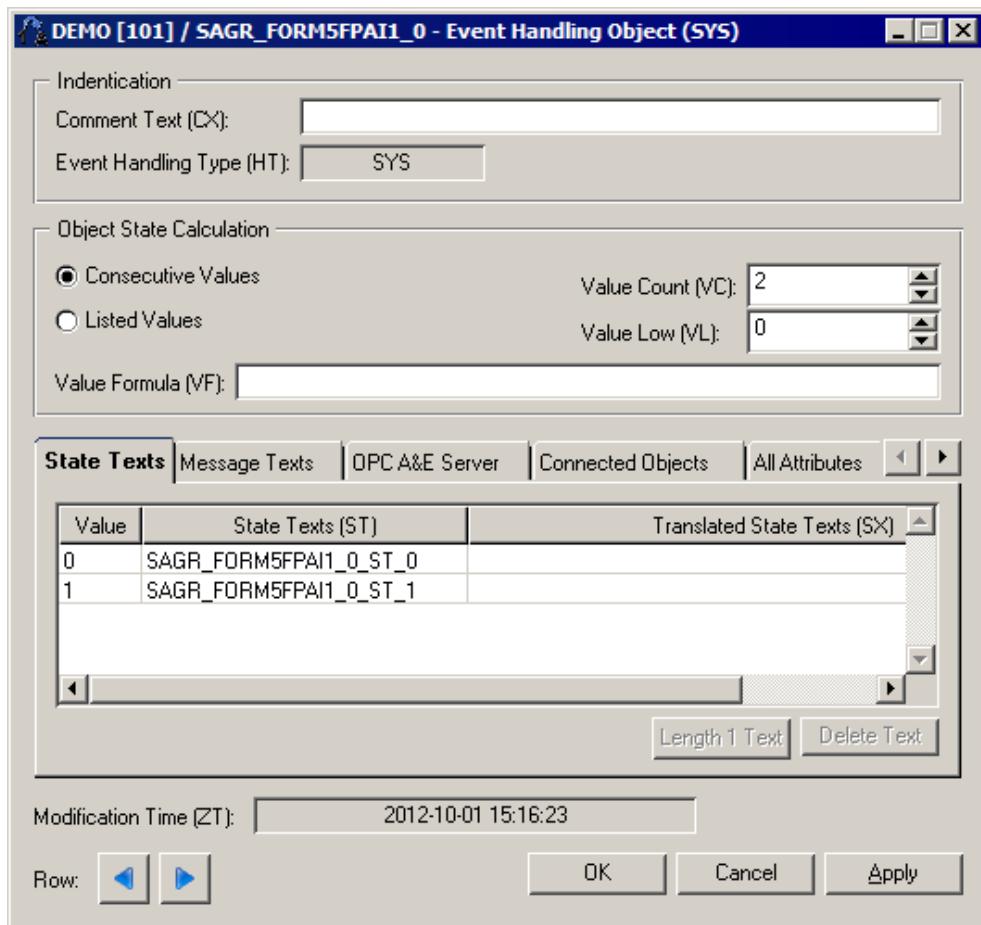


Figure 94: The Event Handling Definition Tool.

18.3.1 Object state calculation area

The Object State Calculation area for the SYS event handling type is shown in Figure 94. See [Section 6.3](#) for the attributes.

18.3.2 OPC A&E event definition

The OPC A&E Event Definition area for the AEC event handling type is shown in Figure 95. See [Section 6.5](#) for the attributes.

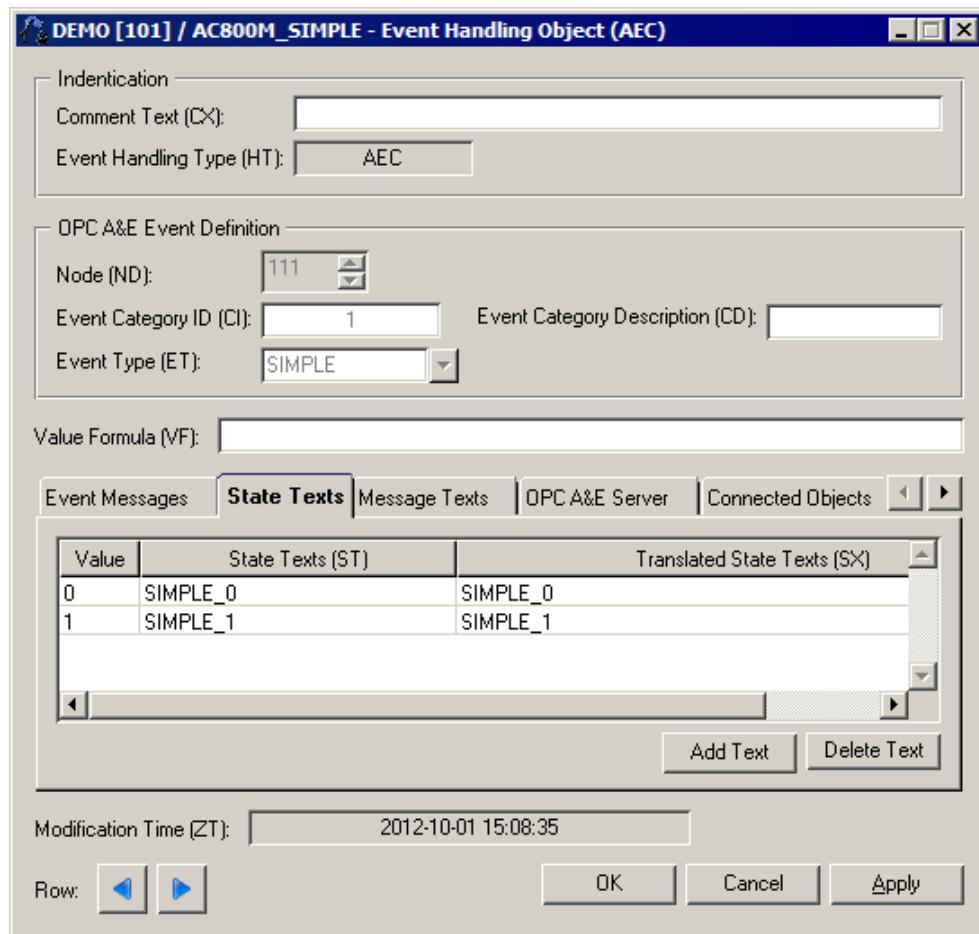


Figure 95: The Event Handling Definition Tool for the event handling of type AEC.

18.4 State and message texts

18.4.1 Overview

Pages **State Texts** and **Message Texts** show state texts (ST) and message texts (MT) and their translation texts SX and MX.

18.4.2 State texts

The **State Texts** page is shown in [Figure 94](#). See [Section 6.3](#) for the attributes.

18.4.3 Message texts

The **Message Texts** page is shown in [Figure 96](#). See [Section 6.3](#) for the attributes.

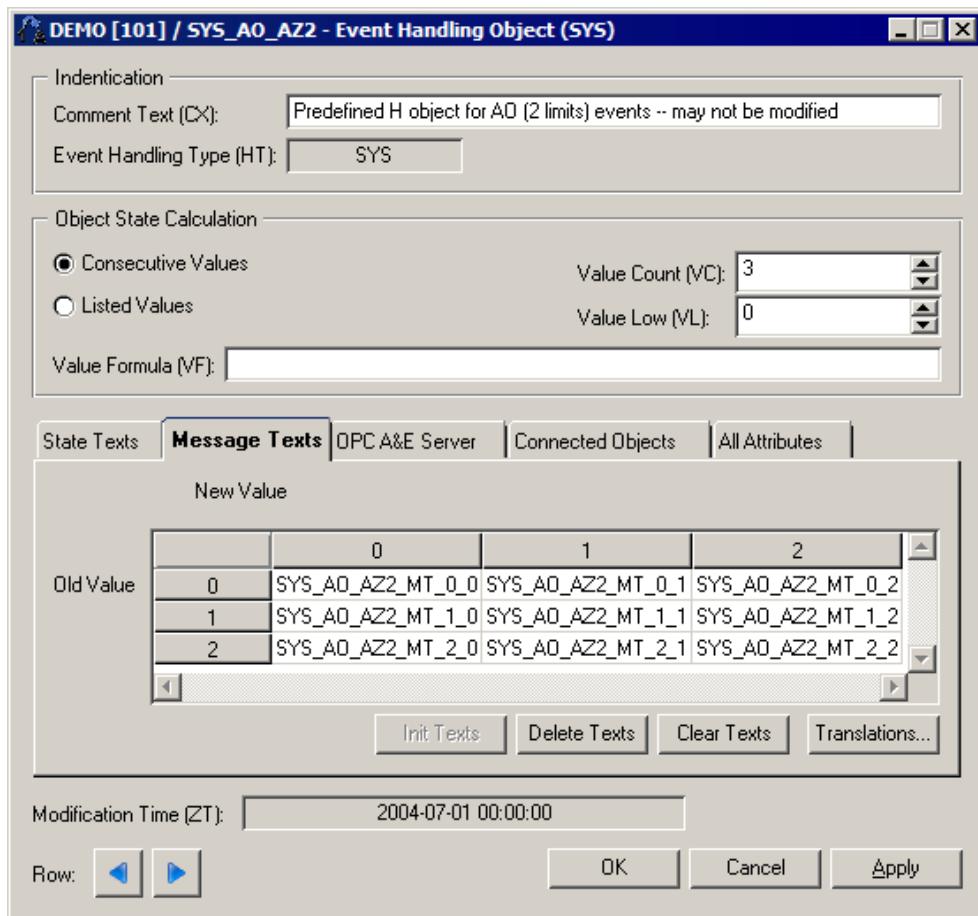


Figure 96: The Message Texts page.

18.5 Event messages and subcondition names

18.5.1 Overview

Pages **Event Messages** and **Subcondition Names** show event messages (EM) and subcondition names (SN).

18.5.2 Event messages

The **Event Messages** page is shown in [Figure 95](#). See [Section 6.5](#) for the attribute.

18.5.3 Subcondition names

The **Subcondition Names** page is shown in [Figure 97](#). See [Section 6.5](#) for the attribute.

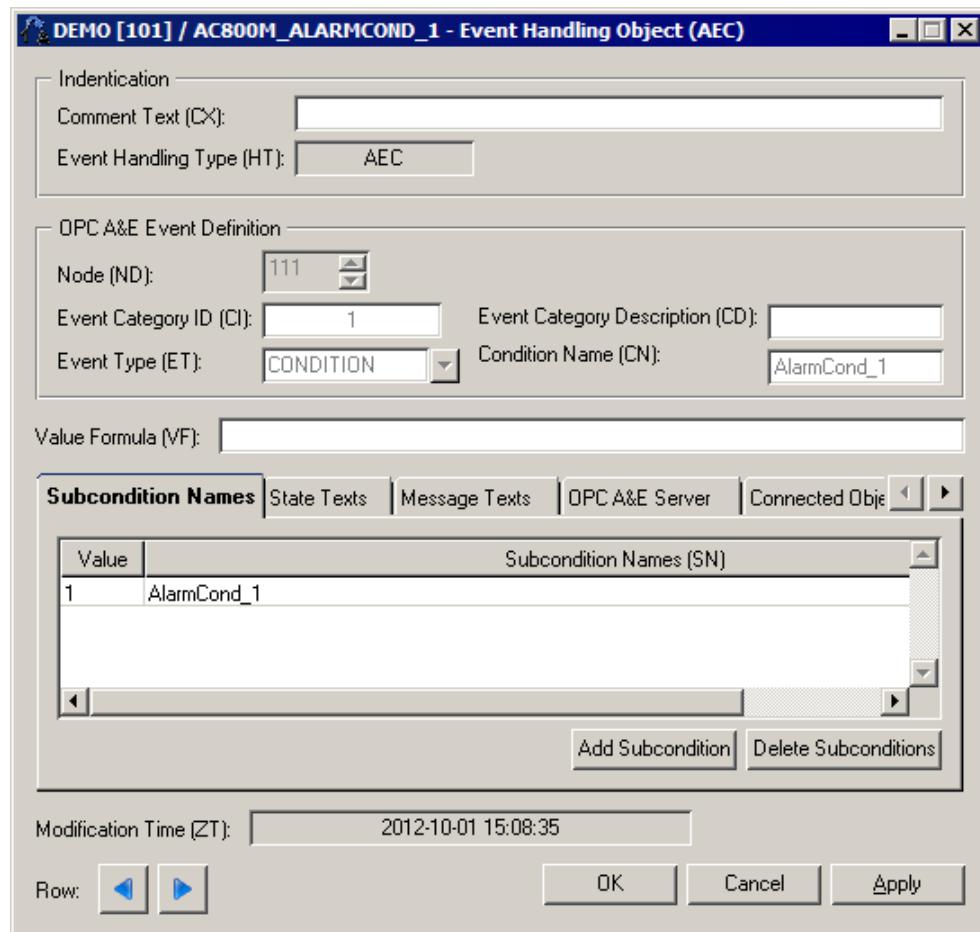


Figure 97: The Subcondition Names page

18.6 OPC A&E server

18.6.1 Overview

The page **OPC A&E Server** shows OPC Alarms & Events server attributes. See [Section 6.4](#) for the attributes.

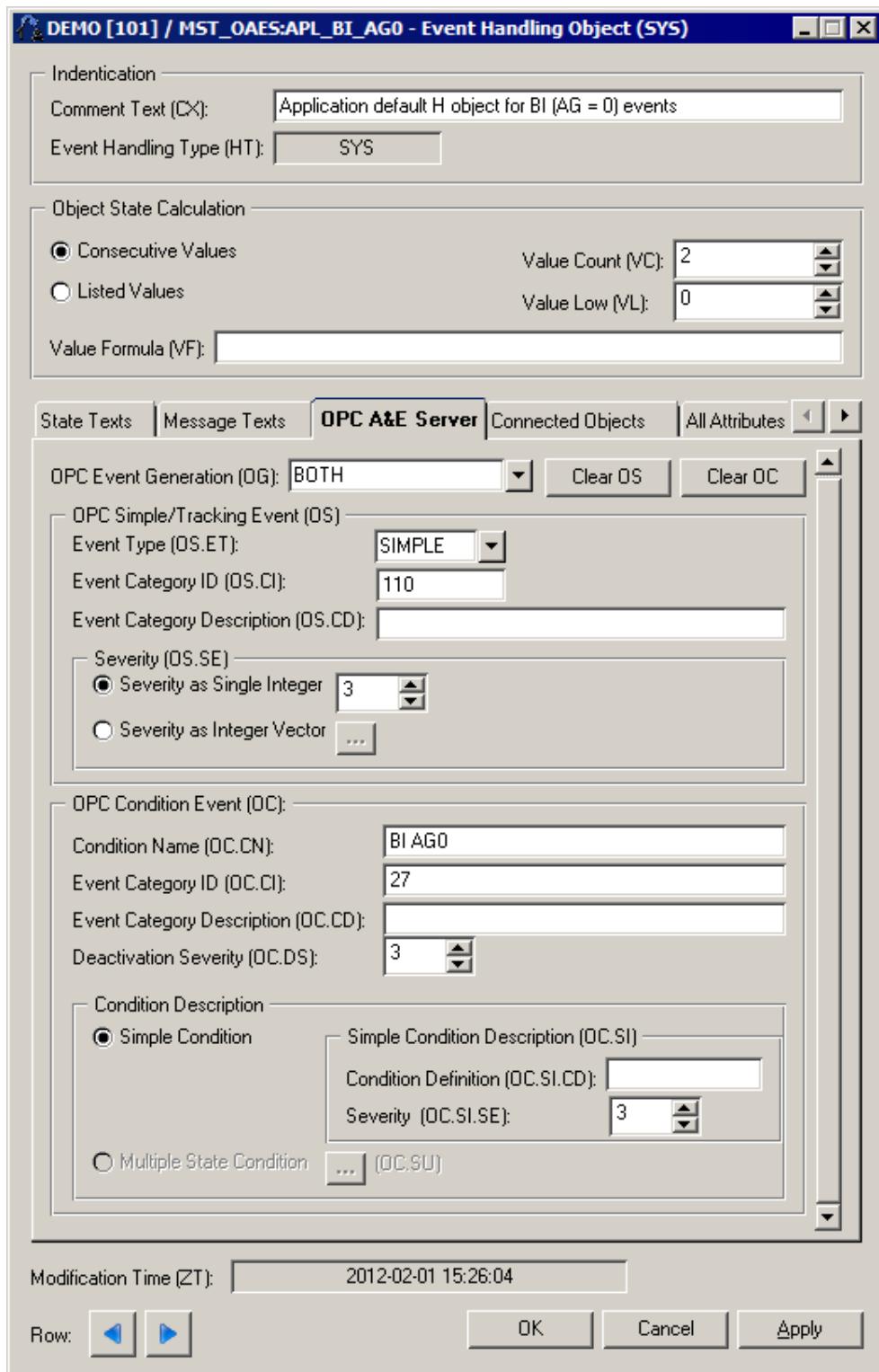


Figure 98: The OPC A&E Server page

18.7 Listed values

If the option **Listed Values** is selected on the Object State Calculation area, the Value Count attribute (VC) is edited by the **Listed Values** dialog (see [Figure 99](#)).

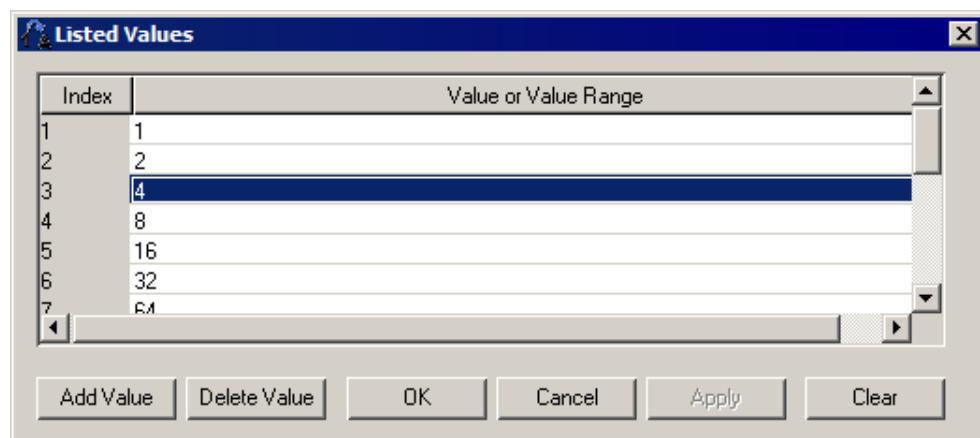


Figure 99: The Listed Values dialog.

Section 19 Scale Object Definition Tool

19.1 About this section

This section presents the definition tool for defining scales and provides a point-by-point description of the data and text boxes in the tool.

19.2 General

The **Object Navigator** has access to the definition tool for defining scales by double-clicking a scale object. A new object is created in the **Object Navigator**. The procedure is described in [Section 16.3](#). The tool is also obtained from the Process Object Definition Tool by selecting to view the scale of a process object.

The definition tool contains a common area and two or three pages. The alternative pages are **Connected objects**, **All Attributes**, **Linear Scaling** and **Stepwise Linear Scaling**. The Scaling Algorithm determines which pages are shown. See [Figure 100](#). The page **Connected objects** lists all process objects (if any) connected to the current scale. The definition of a process object can be shown by selecting a row on the list and pressing **Show** - button or simply double-clicking on a row. The page **All Attributes** contains all attributes in alphabetical order.

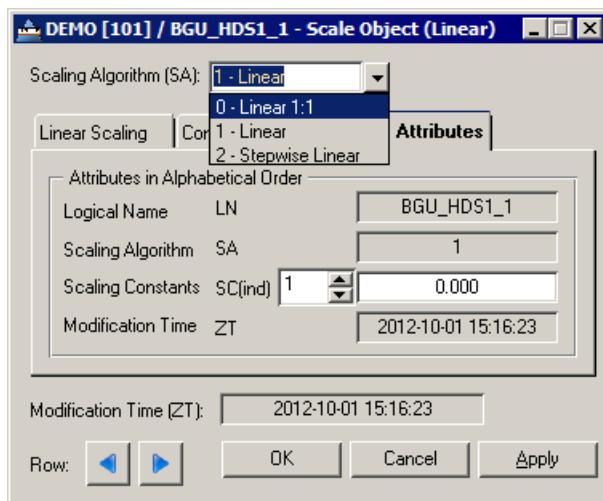


Figure 100: The tool for defining scales. The scaling algorithm one-to-one scaling is selected.

19.3 Common area

The common area above and below the pages contains Scaling Algorithm (SA) and Modification Time (ZT) attributes (see [Figure 100](#)). See [Section 7.3](#) for the attributes.

19.4 Linear scaling

[Figure 101](#) shows the **Linear Scaling** page of the Scale definition tool.

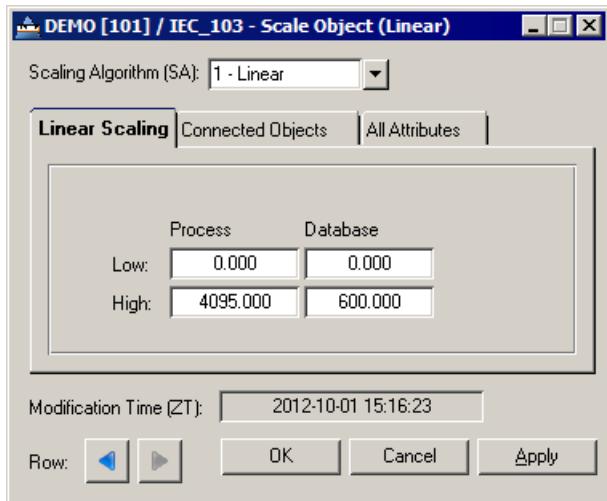


Figure 101: The Linear Scaling page.

19.5 Stepwise linear scaling

[Figure 102](#) shows the page for defining Stepwise Linear Scaling.

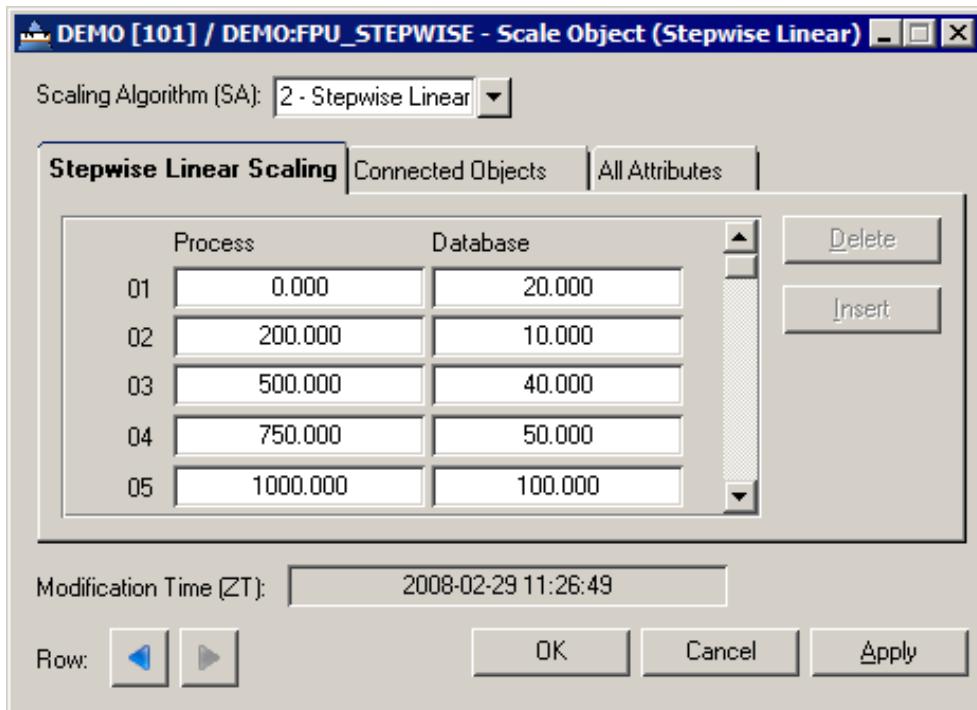


Figure 102: The Stepwise linear page of the scale definition tool.

The page shows 50 pairs of text boxes, each of which correspond to a coordinate on the scaling algorithm curve, [Section 7.3](#).

The **Insert** button allows the user to enter scaling points in-between two points. The **Delete** button deletes the selected point. The insertion and deletion place are determined by clicking one of the text boxes in the correct row. Inserting moves the selected row and all rows below it one row downwards. Deleting removes the selected row and moves all the rows below it one row upwards.

Section 20 Data Object Definition Tool

20.1 About this section

This section describes how to define data objects using the definition tool.

20.2 Overview

Access the **Data Object Definition Tool** from the **Object Navigator** by double-clicking a data object. A new object is created in the **Object Navigator**. The procedure is described in [Section 16.3](#). The tool is also accessed from the definition tools for time channel and event channel.

The **Data Object Definition Tool**, see [Figure 103](#), is composed of a common area and six pages. The pages contain the following definitions and information:

- The **Data Registration** page defines the calculation of the object.
- The **Data** page lists the registered data and provides means for editing the object value, the status code and registration time of certain indices.
- The **Execution Control** page defines the automatic time activation of the data object and the executing tasks.
- The **Storage** page defines the file where the object is saved. It also defines things concerning the registration and storage of data.
- The **External Logging** page specifies the logging profile object to be used when logging values to the SYS600 Historian database(s). It also tells the names of Historian database tags that receive data from this data object.
- The **All Attributes** page lists all attributes with their values in alphabetical order.

To define a new data object or edit an existing one, double-click the object name in the **Object Navigator** and modify the desired attribute values on the data object form. All attributes can be checked in the **All Attributes** page, and view registered data in the **Data** page.

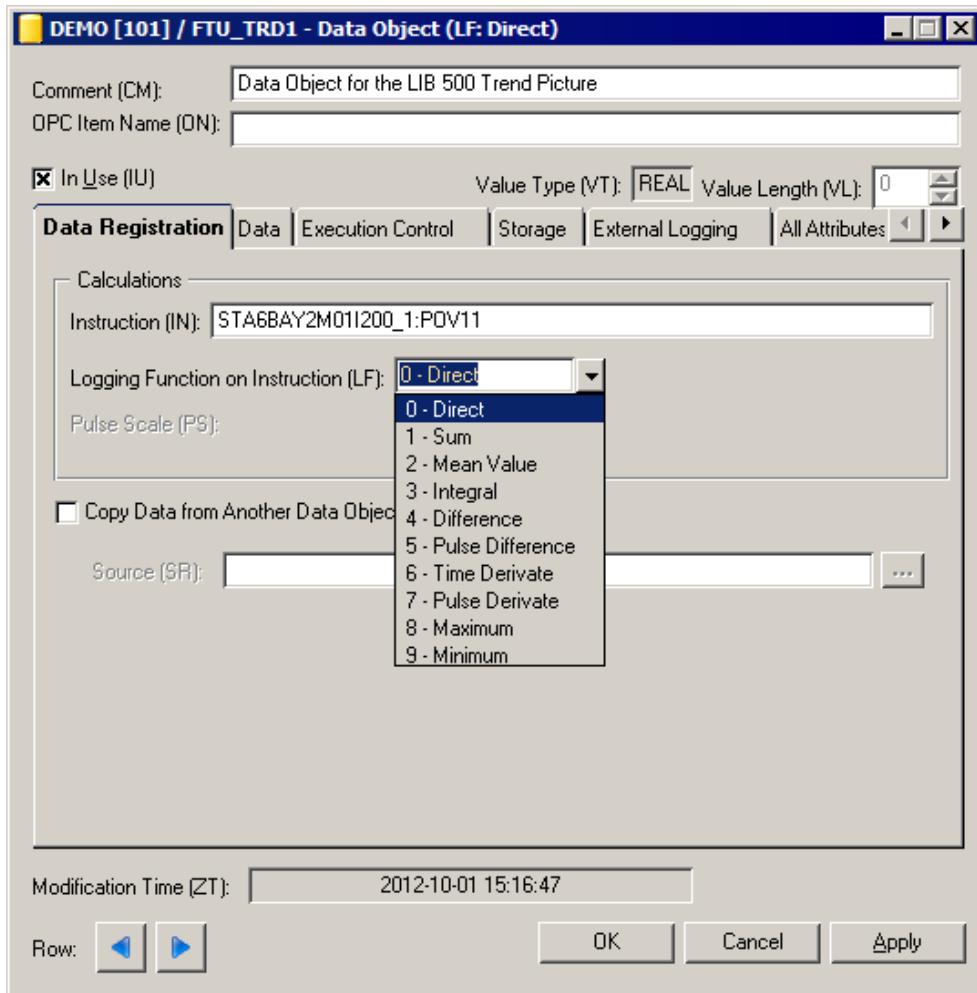


Figure 103: The Data Object Definition Tool includes six pages. In the first one, Data Registration, you can define the calculation of the object.

20.3 Common area

The tool has a common area above and below the pages (see [Figure 103](#)). See [Section 8.2.1](#) for the attributes.

20.4 Data registration

The **Data Registration** page specifies the calculations to be performed at data object execution. The list of SR (Source) shows only Data Objects of the same type (VT).

The Selector dialog can be opened by clicking the  button. See [Figure 103](#). This button is enabled only when the **Copy Data from Another Data Object** checkbox is marked.

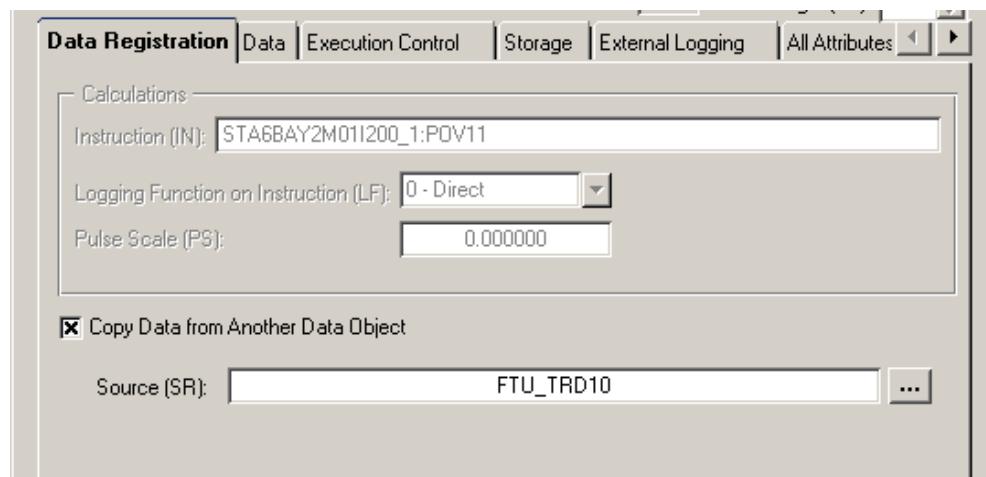


Figure 104: Copy Data from Another Data Object check box is checked to enable logging function COPY

20.5 Data

The second page of the **Data Object Definition Tool** is shown in [Figure 105](#). This page shows the dynamic values of the data objects **value (OV)**, **status (OS)** and registration time (RT). See [Section 8.2.6](#) for the attributes.

In the area showing the registered values, the indices are shown to the left. Browse through the indices using the scroll bars. Above the registered values is the internal object value (stored value) with status codes and registration time.

The tool reads the values from the report database. The shown values can be updated by clicking the **Fetch** button in the page. There are also possibilities for listing the registered values. The listing order and the listed indices can be defined using the controls to the right from the Registered Values list. Double arrow buttons can be used to browse registered values. The dynamic attributes in the tool can be changed.

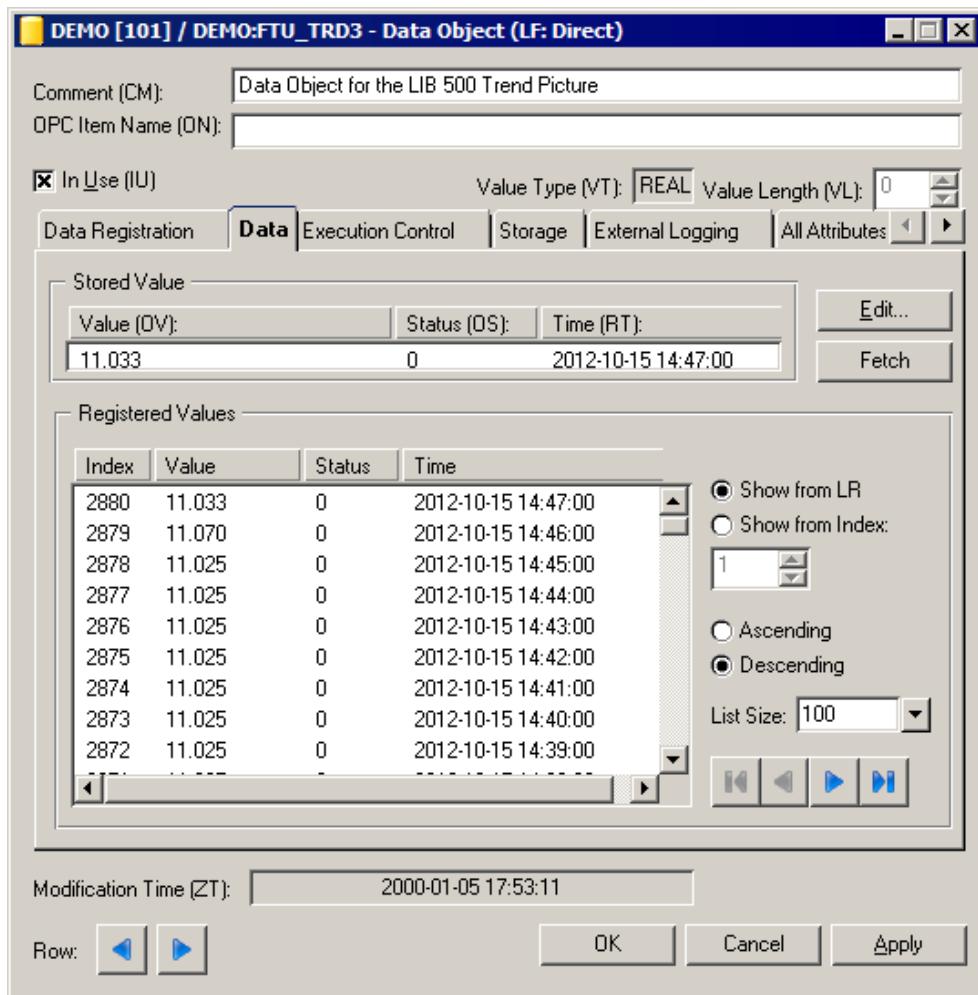


Figure 105: The second page of the Data Object Definition Tool contains dynamic data of the Data Objects. You can view or change the data.

To change the registered attributes or the internal value:

1. Click the data post to be edited.
2. Click the **Edit...** button.
3. The dialog in [Figure 106](#) is shown on the screen. The upper row shows the old values of the attributes. Type the new values in the lower text box row.
4. Click **OK** to exit the dialog and confirm the changes or **Cancel** to exit without saving the changes.



The value is updated directly to the database immediately when **OK** is clicked. This means that the changes cannot be canceled by clicking **Cancel** in the definition tool.

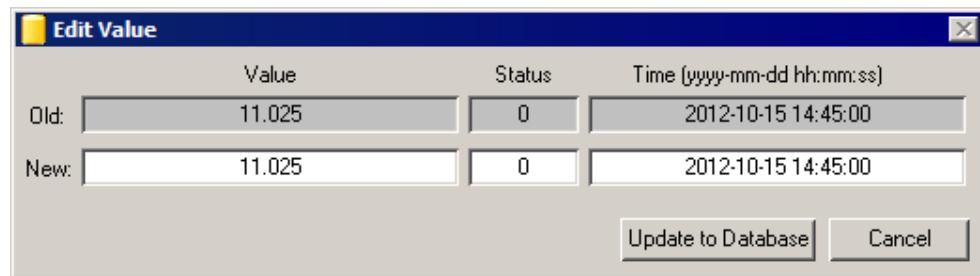


Figure 106: Using this dialog you can change the values of OV, OS and RT attributes.

20.6 Execution control

[Figure 107](#) shows the **Execution Control** page of the **Data Object Definition Tool**. In this page, the time activation and the executing tasks can be specified. See [Section 8.2.4](#) for the attributes.

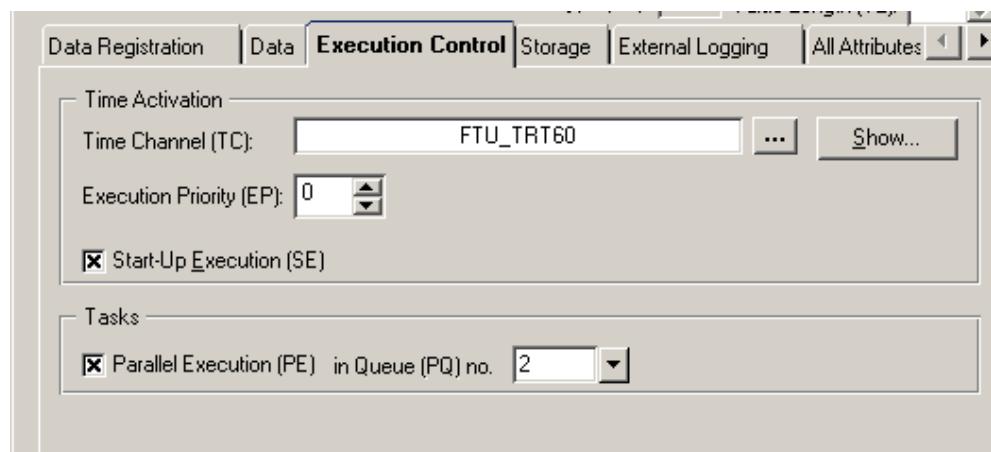


Figure 107: The Execution Control page of the Data Object Definition Tool

20.7 Storage

[Figure 108](#) shows the **Storage** page of the **Data Object Definition Tool**. See [Section 8.2.5](#) for the attributes.

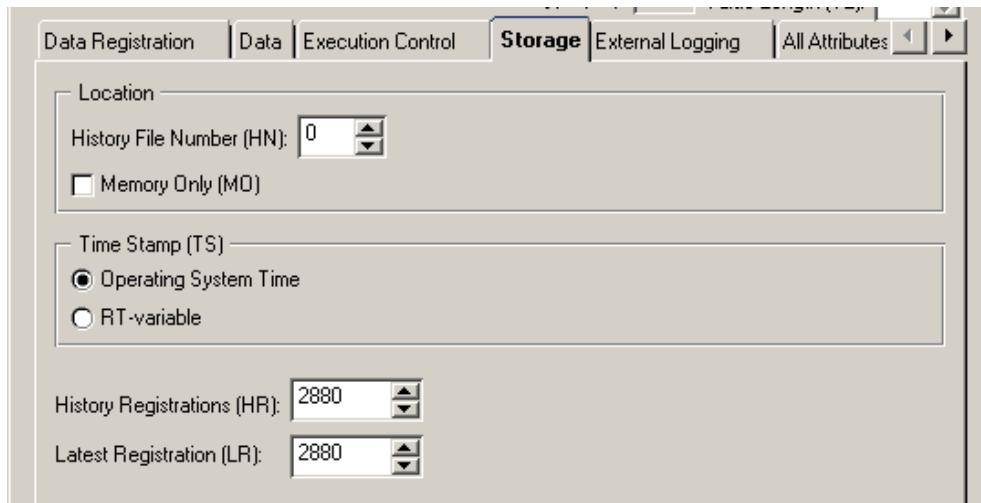


Figure 108: The storage page of the Data Object Definition Tool

20.8 External logging

Figure 109 shows the **External Logging** page of the **Data Object Definition Tool**. See [Section 8.2.5](#) for the attributes.

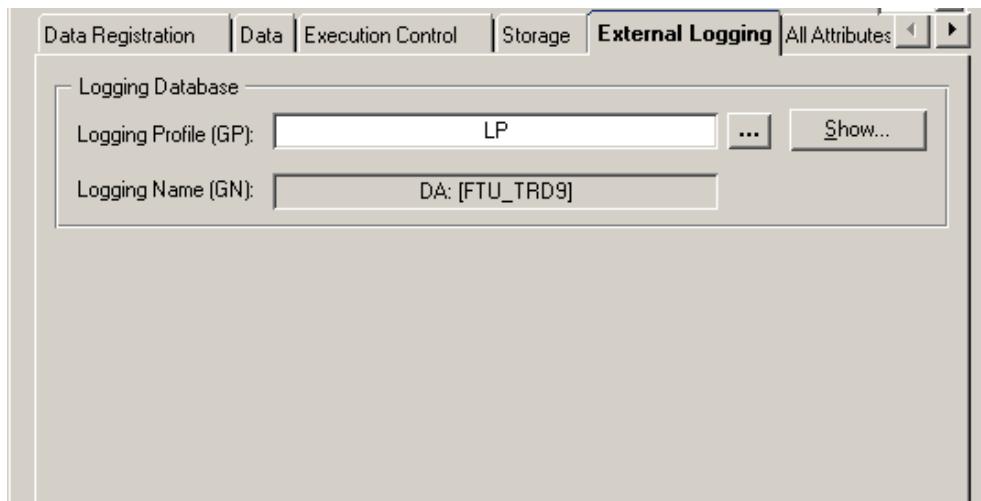


Figure 109: The external logging page of the Data Object Definition Tool

20.9 All attributes

The page named **All Attributes** is shown in Figure 110. The page lists all attributes in alphabetical order. The attribute values in white text boxes can be edited. The attribute values with grey background cannot be edited.

Attributes in Alphabetical Order		
Comment	CM	Data Object for the LIB 500 Trend Picture
Execution Priority	EP	0
Free Integer	FI	0
Free Text	FX	
Logging Name	GN	DA: [FTU_TRD9]
Logging Profile	GP	LP
History File Number	HN	0
History Registrations	HR	2880
Instruction	IN	STA6BAY2M02U204:POV16
In Use	IU	1
Logging Function	LF	0
Logical Name	LN	FTU_TRD9
Latest Registration	LR	2880
Memory Only	MO	0
OPC Item Name	ON	
Object Status	OS	10
Object Status Indexes	OS(Ind)	1 10
Object Value	OV	
Object Value Indexes	OV(Ind)	1
Parallel Execution	PE	0
Parallel Queue	PQ	0
Pulse Scale	PS	0.000000
Registration Time	RT	2012-10-16 08:35:30
Registration Time Indexes	RT(Ind)	1 2012-10-15 08:36:00
Start-up Execution	SE	0
Source	SR	
Time Channel	TC	FTU_TRT30
Time Stamp	TS	0

Figure 110: The All Attributes page of the Data Object Definition Tool

Section 21 Command Procedure Definition Tool

This section describes how to define command procedures using the definition tools.

21.1 Overview

The definition tool for command procedures is accessed from the Object Navigator by double-clicking a command procedure. A new object is created in the Object Navigator. The procedure is described in [Section 16.3](#). It can also be accessed from time channel and event channel tools.

The **Command Procedure Definition Tool** comprises a common area and four pages. The pages contain the following definitions and information:

- The **Procedure** page contains the program of the command procedure.
- The page **Execution Control** defines the automatic time activation of the command procedure and the executing tasks.
- The **Storage** page defines the file where the object is saved and the origin of the time stamp.
- The **All Attributes** page finally lists all attributes in alphabetical order.

21.2 Common area

The tool has a common area above and below the pages (see [Figure 111](#)). See [Section 9.2.1](#) for the attributes.

21.3 Procedure page

[Figure 111](#) shows the **Procedure** page of the **Command Procedure Definition Tool**. The Instruction (IN): list contains the program.

To type a new program or edit an existing one:

1. Click the **Edit** button or double-click the Instruction (IN): list.
2. The SCIL editor appears as a separate dialog. The general functions of the SCIL Editor are described in the Programming Language SCIL manual. It is also possible to import and export text from the SCIL Program Editor. To import, choose **Import** and to export, choose **Export** from the **File** menu. The import function opens a file chooser, where the name of the file to be imported is specified. The contents of the file is loaded in the SCIL Program Editor when **OK** or **Apply** is clicked. It is placed starting from the row where the cursor is, if the row is empty. If there is text in the row, the contents is placed starting from the next row. The Export function also opens a file chooser where the name of the file is specified. If no text is selected, everything in the SCIL Program Editor is transferred into the specified file. Note that if an existing file is specified, export overwrites the previous contents of the file.

Compile IN when Edited

If check box is checked, the compilation is done always when user clicks **Apply** or **OK** button. A successful compilation updates the IN and CP-attributes.

Compiling a Command Procedure:

1. Click **Edit**. The SCIL editor is opened. Enter the Command Procedure or edit an old one. If the Compile IN when Edited procedure mentioned above has been done, the following step 2 can be skipped. Otherwise, proceed to step 2.
2. In the SCIL editor, check the **Compilation in Use** option on the **Options** menu.
3. Choose **Update** on the **File** menu.
4. Choose **Exit** on the **File** menu.

Compile IN when Edited is checked, because compilation was taken into use in the SCIL editor. A successful compilation updates the IN and CP-attributes and 1 (Compiled) is shown in the **Compilation State (CS)** field.

If compilation is cancelled in the SCIL editor (IN could not be compiled or user cancelled the compilation), the IN attribute is updated and the CP attribute is emptied.

If compiled program exists, the **Uncompile** button can be used to clear the CP attribute.

If the command procedure is compiled (CP not empty), the Command Procedure tool automatically checks the check box on **Procedure** tab. Otherwise (not compiled), the check box is not checked as default.

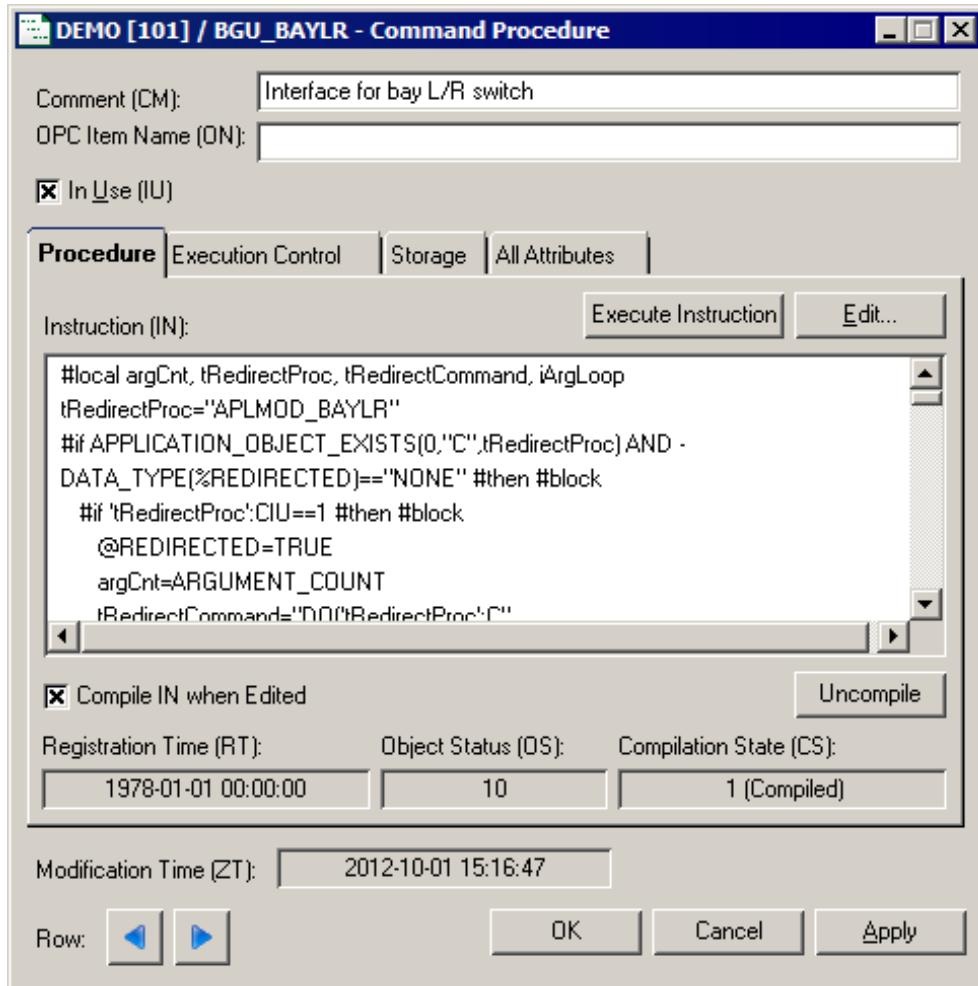


Figure 111: The Procedure page of the Command Procedure Definition Tool

21.4

Execution control page

Figure 112 shows the **Execution Control** page. See [Section 8.2.4](#) for the attributes.

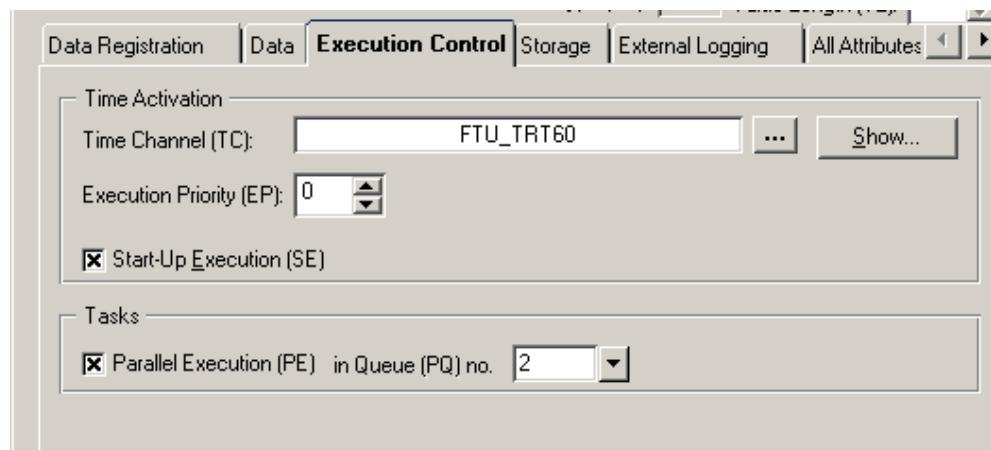


Figure 112: The Execution Control page of the Command Procedure Definition Tool

21.5 Storage page

Figure 113 shows the **Storage** page of the definition tool. See [Section 9.2.5](#) for the attributes.

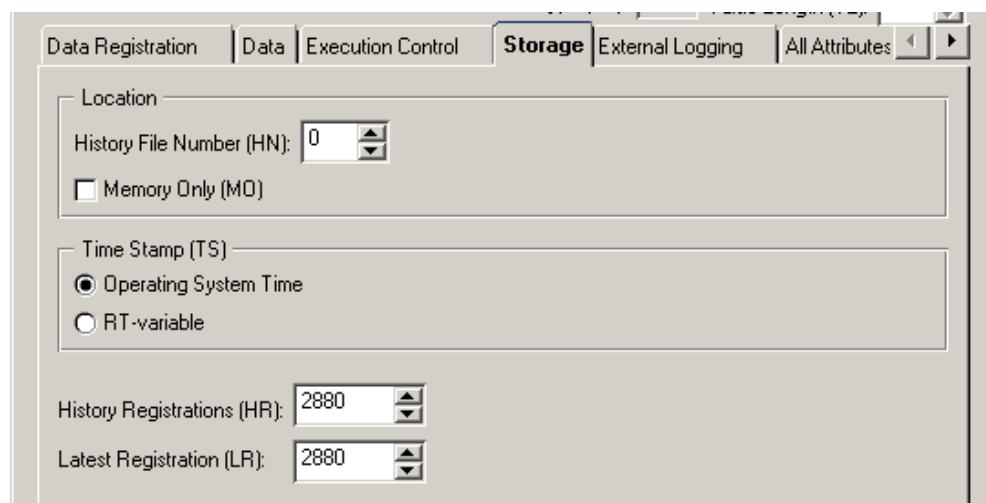


Figure 113: The Storage page of the Command Procedure Definition Tool

21.6 All attributes

Figure 114 shows the page **All Attributes**. This page lists all command procedure attributes in alphabetical order. The attribute values in white text boxes can be edited, whereas attribute values with grey background cannot be edited.

Attributes in Alphabetical Order		
Comment	CM	Data Object for the LIB 500 Trend Picture
Execution Priority	EP	0
Free Integer	FI	0
Free Text	FX	
Logging Name	GN	DA: [FTU_TRD9]
Logging Profile	GP	LP
History File Number	HN	0
History Registrations	HR	2880
Instruction	IN	STA6BAY2M02U204:POV16
In Use	IU	1
Logging Function	LF	0
Logical Name	LN	FTU_TRD9
Latest Registration	LR	2880
Memory Only	MO	0
OPC Item Name	ON	
Object Status	OS	10
Object Status Indexes	OS(Ind)	1 10
Object Value	OV	
Object Value Indexes	OV(Ind)	1
Parallel Execution	PE	0
Parallel Queue	PQ	0
Pulse Scale	PS	0.000000
Registration Time	RT	2012-10-16 08:35:30
Registration Time Indexes	RT(Ind)	1 2012-10-15 08:36:00
Start-up Execution	SE	0
Source	SR	
Time Channel	TC	FTU_TRT30
Time Stamp	TS	0

Figure 114: All Attributes page of the Command Procedure Definition Tool

Section 22 Time Channel Definition Tool

22.1 About this section

This section provides a point-by-point description of the input fields and function buttons in the **Time Channel Definition Tool**.

22.2 Overview

The definition tool for time channels is accessed from the **Object Navigator** by double-clicking a time channel. A new object is created in the **Object Navigator**. The procedure is described in [Section 16.3](#). It can also be accessed from data object and command procedure tools.

The **Time Channel Definition Tool**, see [Figure 115](#), comprises a common area and five pages containing the following definitions and information:

- The **Execution** page specifies the Execution of the time channel.
- The **Initialization** page specifies the Initialization of the time channel.
- The **Execution Control** page defines the execution policies and the executing tasks.
- The **Connected objects** page shows the objects connected to the time channel.
- The **All Attributes** page finally lists all attributes in alphabetical order.

22.3 Common area

The tool has a common area above and below the pages (see [Figure 115](#)). See [Section 10.1](#) for the attributes.

22.4 Execution

The **Execution** page specifies the execution time of the time channel. See [Section 10.1](#) for an explanation of the execution of time channels and [Section 10.2.2](#) for the attributes.

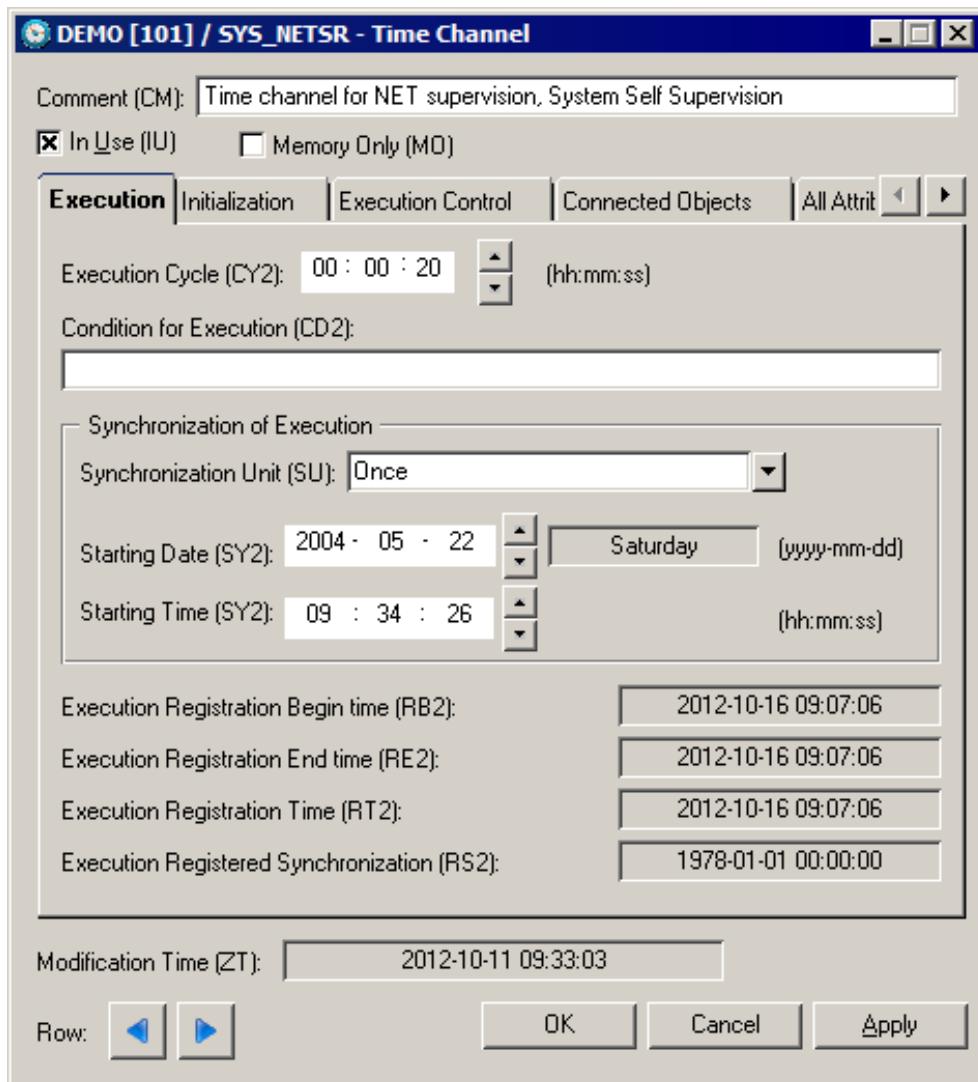


Figure 115: The Execution page of the Time Channel Definition Tool

22.5 Initialization

Figure 116 shows the **Initialization** page. This page specifies the initialization times of the time channel. See [Section 10.1](#) for an explanation of what initialization means and [Section 10.2.2](#) for the attributes.

Execution cycle The time interval for periodically recurrent executions if such is desired. The fields can be left empty. The CY attribute, index 2.

Condition for ... A conditional expression according to the rules of SCIL. Enter a condition if you wish to limit the occurrences of executions. Initialization/execution occurs only when the condition is fulfilled. See, the CD attribute.

Synchronization The synchronization times. The selections are a combination of SU and SY attributes. The synchronization alternatives are the same as for execution.

The time settings farthest down on the page show the latest initialization and synchronization times.

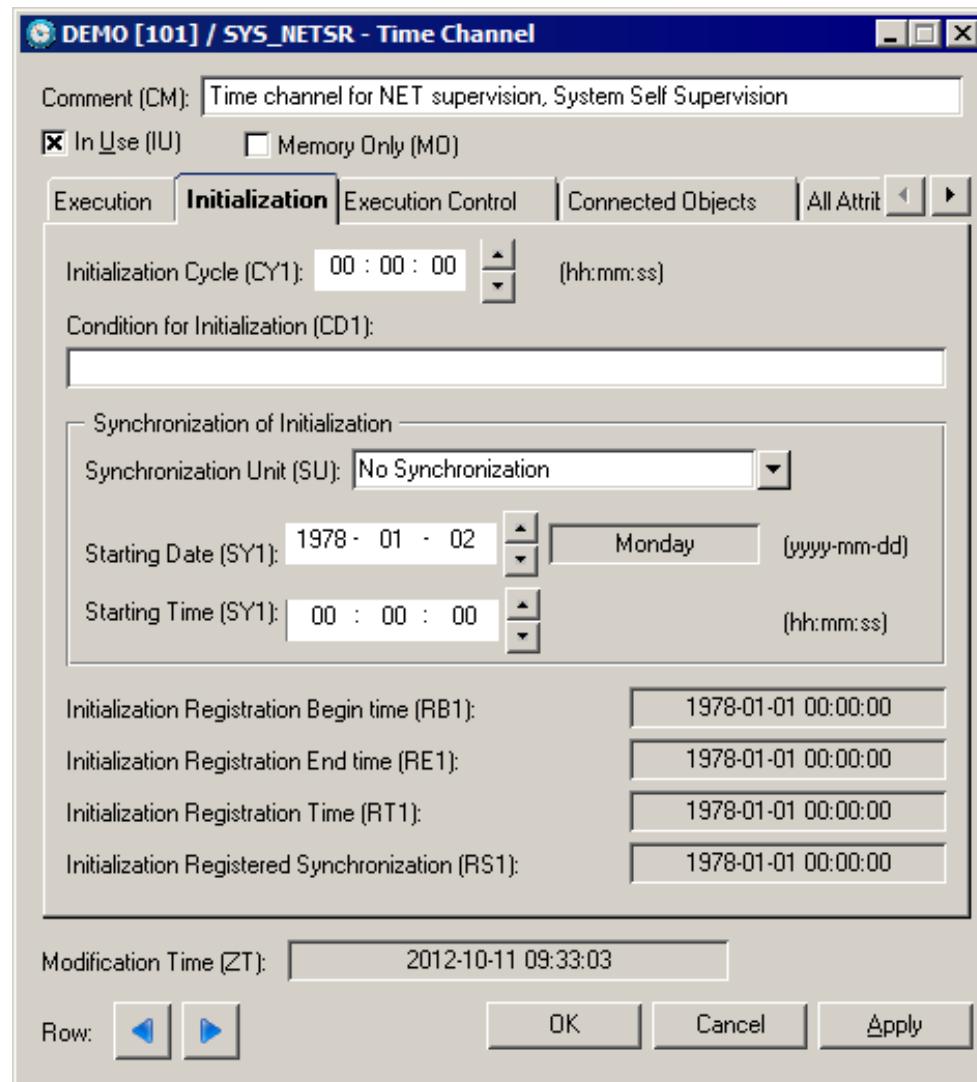


Figure 116: The Initialization page of the Time Channel Definition Tool

22.6 Execution control

Figure 117 shows the **Execution Control** page. See [Section 10.2.2](#) and [Section 10.3](#) for the attributes.

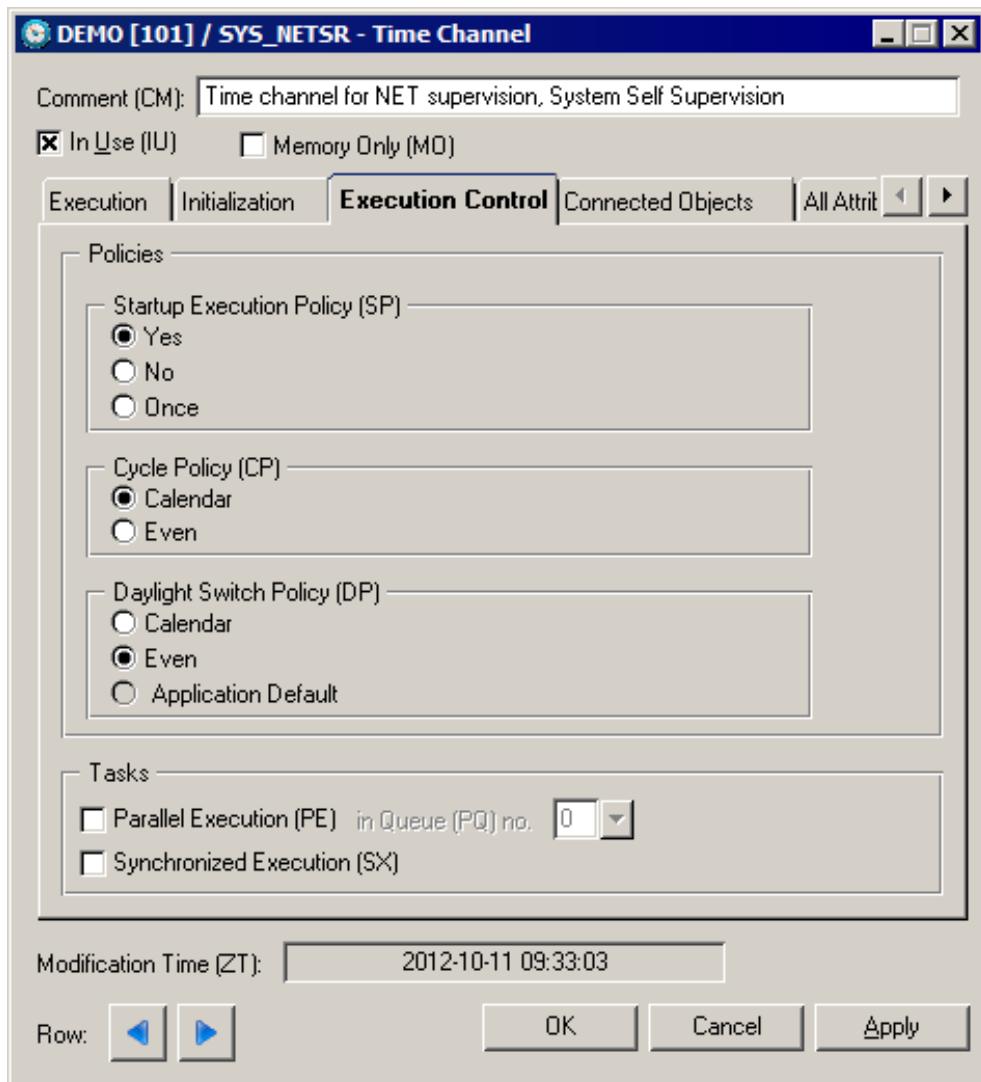


Figure 117: The Execution Control page of the Time Channel Definition Tool

22.7 Connected objects

The **Connected objects** page provides an overview of all data objects and command procedures connected to the time channel (see [Figure 118](#)). Only the first 10 000 objects are shown in the list. If there are more than 10 000 connections, the user is informed.

Each row in the list contains the name of an object, the priority within the time channel and the object type. The page is of informative character and the list of connected objects cannot be edited here. However, the object definitions are accessed from the list. Objects are added to the list when they are defined in their respective tools to be activated by the time channel in question, or by clicking **Add...** button on the page. See [Section 16.4](#) for how to connect several objects in one go.

To view or edit any of the objects in the list, click the object and then **Show...**button.

The definition of the selected object appears in a new window.

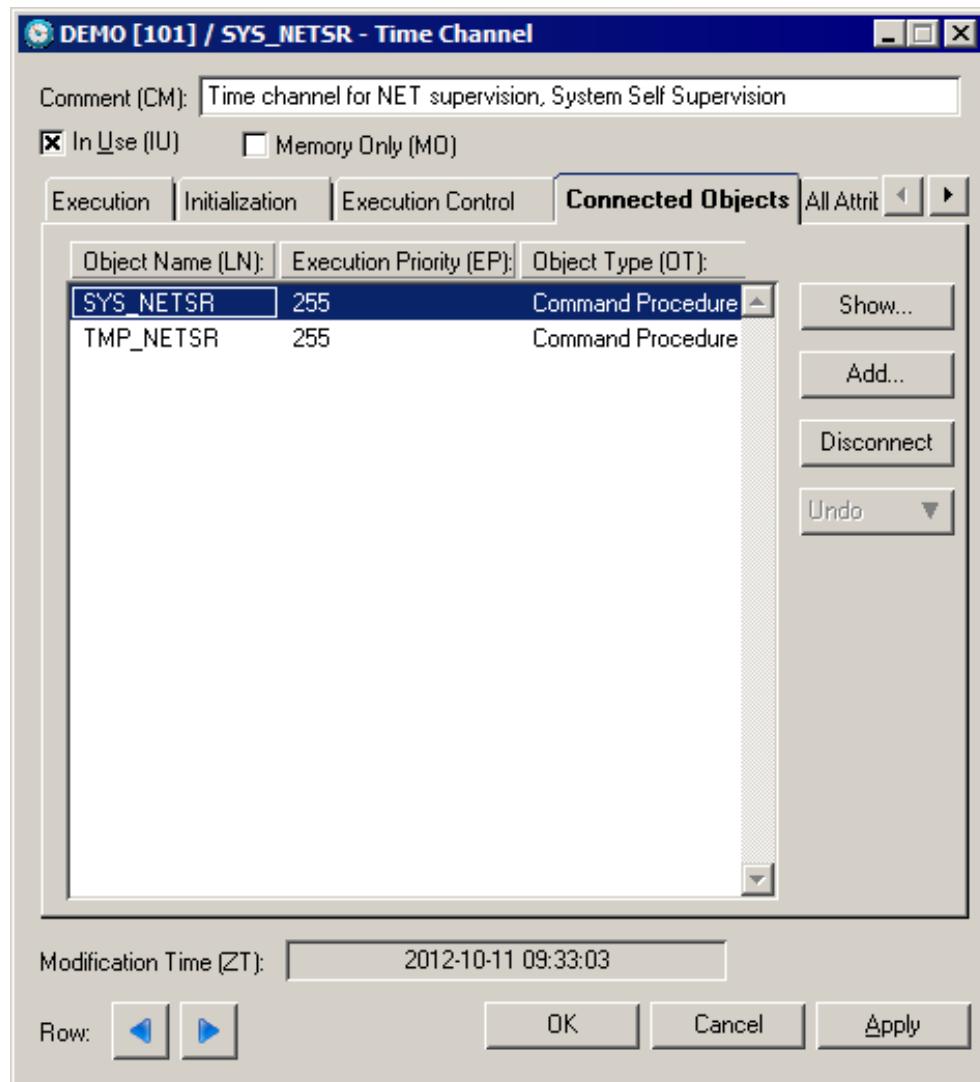


Figure 118: The Connected Objects page of the Time Channel Definition Tool

22.8 All attributes

The **All Attributes** page (see [Figure 119](#)) lists all time channel attributes in alphabetical order. All attributes, except read-only attributes, can be changed. The data fields of the read-only attributes are grey.

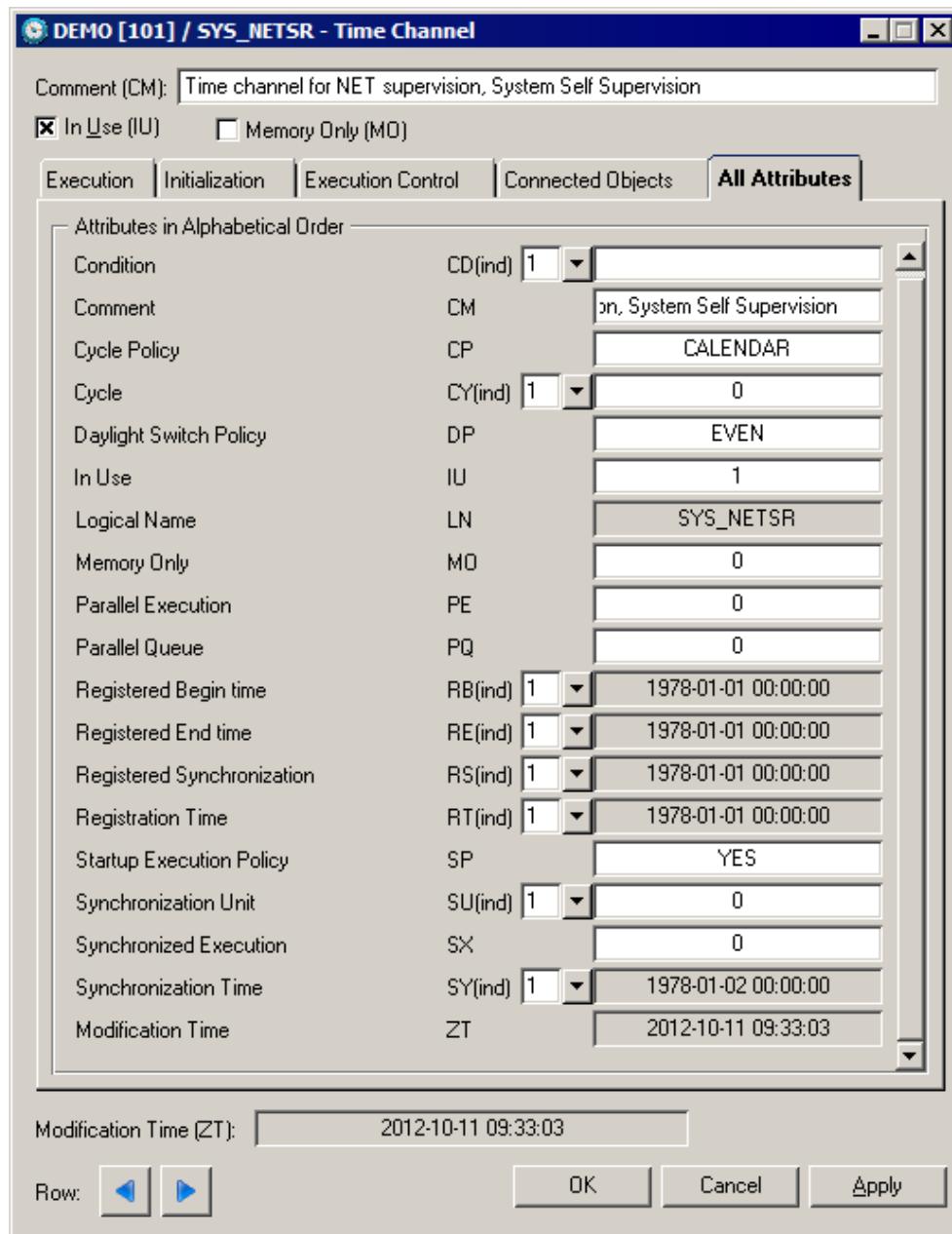


Figure 119: The All Attributes page of the Time Channel Definition Tool

Section 23 Event Channel Definition Tool

23.1 About this section

This section describes how to define event channels using the object definition tools.

23.2 Overview

The definition tool for defining event channels is accessed from the **Object Navigator** by double-clicking an event channel. A new object is created in the **Object Navigator**. The procedure is described in [Section 16.3](#). The tool is also obtained from the **Process Object Definition Tool** by selecting to view the event channel of a process object.

The **Event Channel Definition Tool**, see [Figure 120](#), comprises a common area and three pages containing the following definitions and information:

- The page **Activated Objects** specifies the primary and secondary objects activated by the event channel.
- The page **Connected objects** shows the process object indexes connected to the event channel.
- The page **All Attributes** lists all attributes in alphabetical order.

23.3 Common area

The tool has a common area above and below the pages (see [Figure 120](#)). See [Section 11.2](#) for the attributes.

23.4 Activated objects

[Figure 120](#) shows the **Activated Objects** page. This page specifies the data objects and command procedures activated by the event channel.

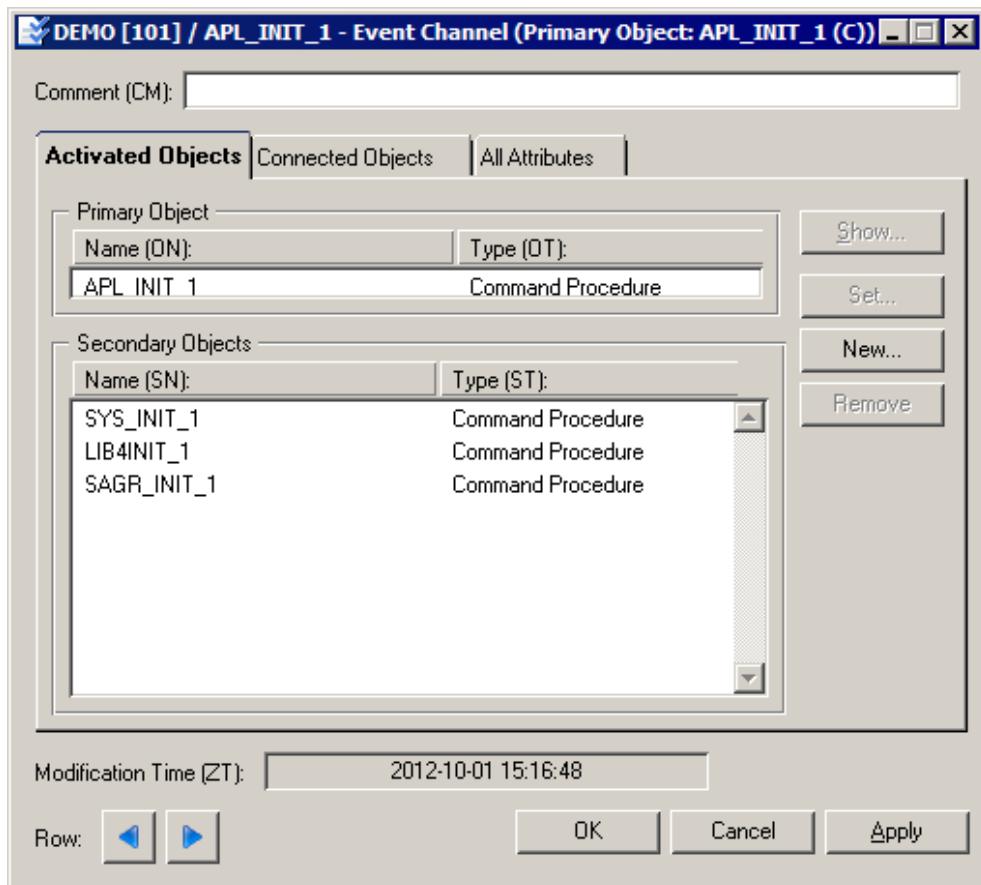


Figure 120: Activated Objects page of an Event Channel

To view the object definitions of the activated objects, click the object name in the list and then click the **Show...** button.

To add a new secondary object, click the **New** button.

To edit an object name or type in the list, click the object name and the **Set** button.

To remove a secondary object from the list, click the object name and **Remove**.

23.5 Connected objects

The **Connected objects** page (see [Figure 121](#)) lists all process objects that activate the event channel.

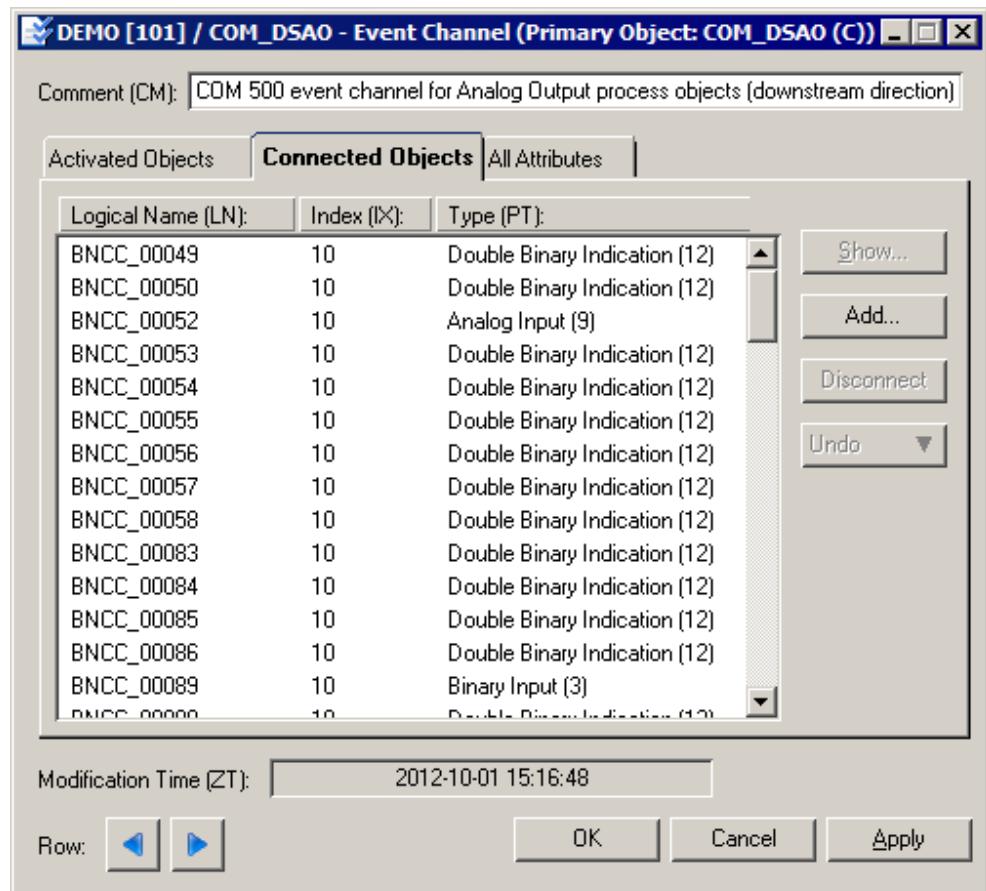


Figure 121: The Connected Objects page

The connection of process objects to the event channel is done in the process object definition or by clicking **Add...** button. See [Section 16.4](#) how to connect several objects in one go. To view or edit the process object definitions, click the object in the list and then click the **Show...** button. The **Show...** button opens the Process Object Definition Tool where the selected objects can be edited as described in [Section 17](#).

23.6 Attribute list

This page (see [Figure 122](#)) lists all event channel attributes in alphabetical order. The attribute values in white text boxes can be edited. The attribute values with grey background cannot be edited.

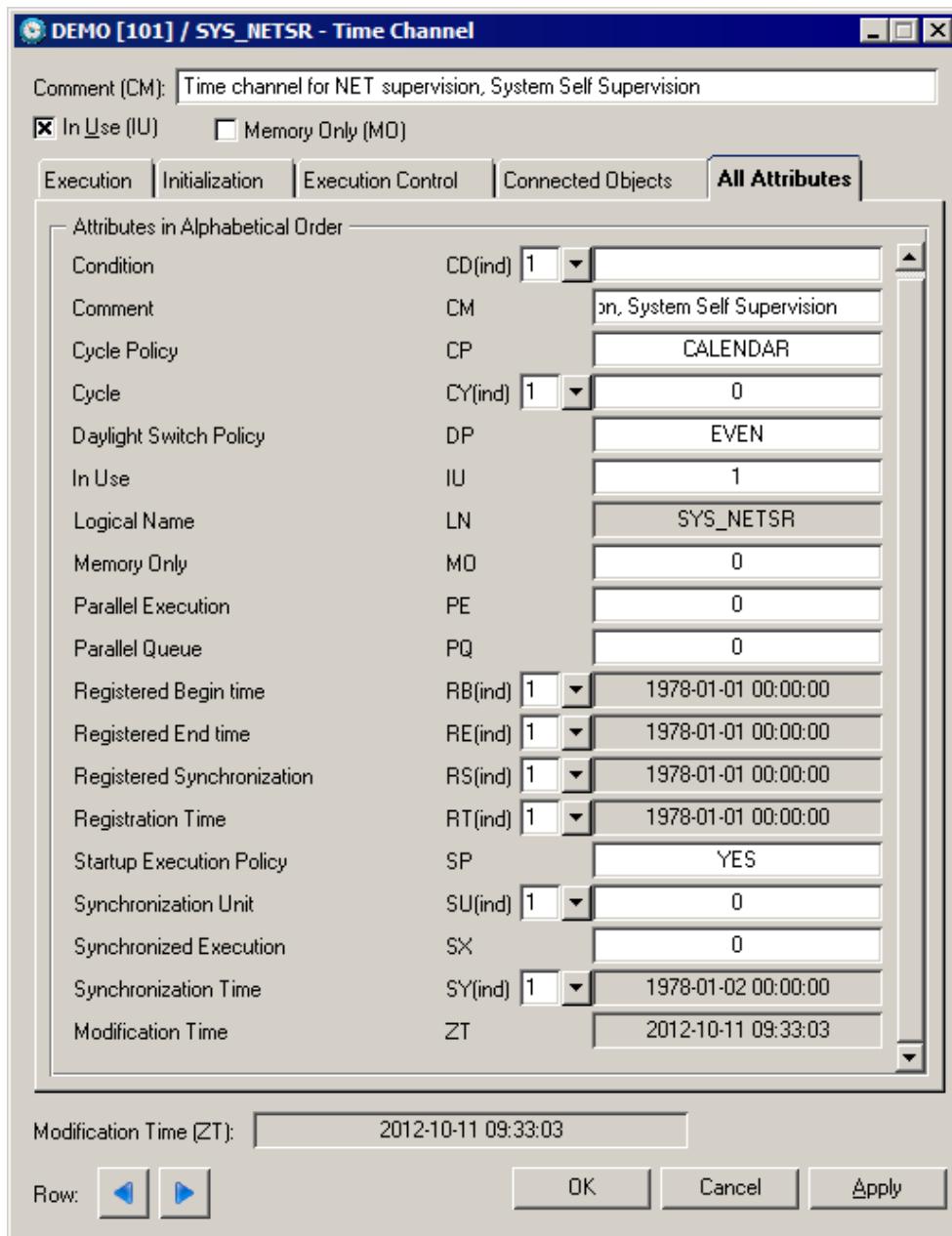


Figure 122: All event channel attributes are listed in alphabetical order

Section 24 Logging Profile Object Definition Tool

24.1 About this section

This section presents the definition tool for defining logging profile objects.

24.2 General

The **Object Navigator** has access to the tool for defining logging profile objects by double-clicking a logging profile object. A new object is created in the **Object Navigator**. The procedure is described in the [Section 16.3](#).

The tool has the common area and two pages for the logging profile of type OBJECT: **Connected objects** and **All Attributes**. The page **Connected objects** lists all process and data objects that are connected to the current OBJECT logging profile. Connecting and disconnecting objects is described in the [Section 16.4](#).

The tool has the common area and three pages for the logging profile of type DATABASE: **Connected objects/Diagnostic Counters** and **All Attributes**. The page **Connected objects** lists all logging profile objects that are connected to the logging profile. The **Diagnostic Counters** page shows counters of the Historian database connection.

The tool has the common area and two pages for the logging profile object of type HISTORY: **Connected objects** and **All Attributes**. The page **Connected objects** lists all logging profile objects that are connected to the logging profile.

The page **All Attributes** contains all attributes that are relevant to the current logging profile object. The attributes are in alphabetical order.

24.3 Definition tool of the logging profile object of type OBJECT

The definition tool of the **Logging Profile Object** of type DATABASE is shown in [Figure 123](#). See [Section 12.2](#) for the attributes.

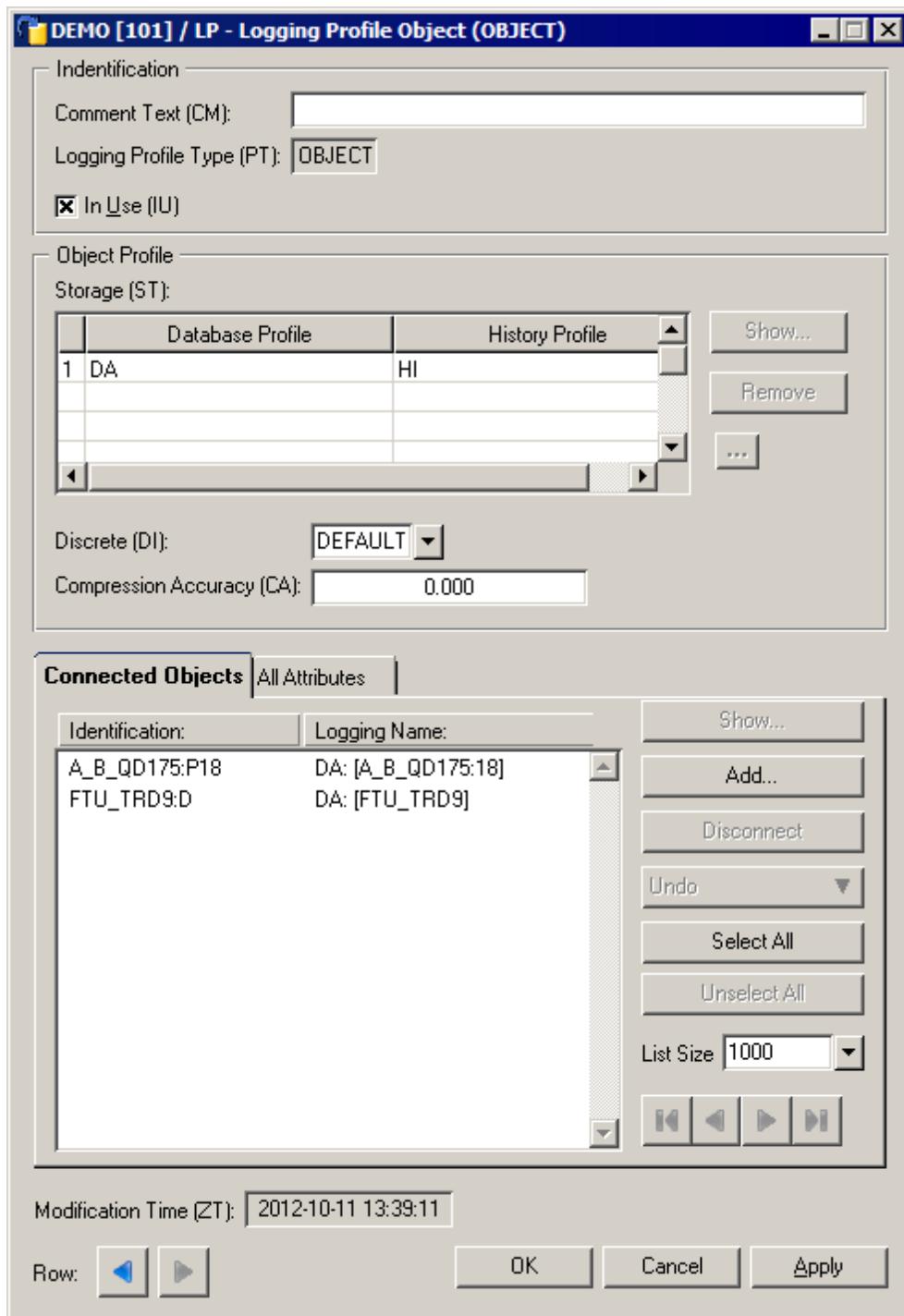


Figure 123: The Definition Tool of the Logging Profile Object of type OBJECT

24.4 Definition tool for logging profile object of type DATABASE

The definition tool of the **Logging Profile Object** of type OBJECT is shown in [Figure 124](#). See [Section 12.2](#) for the attributes.

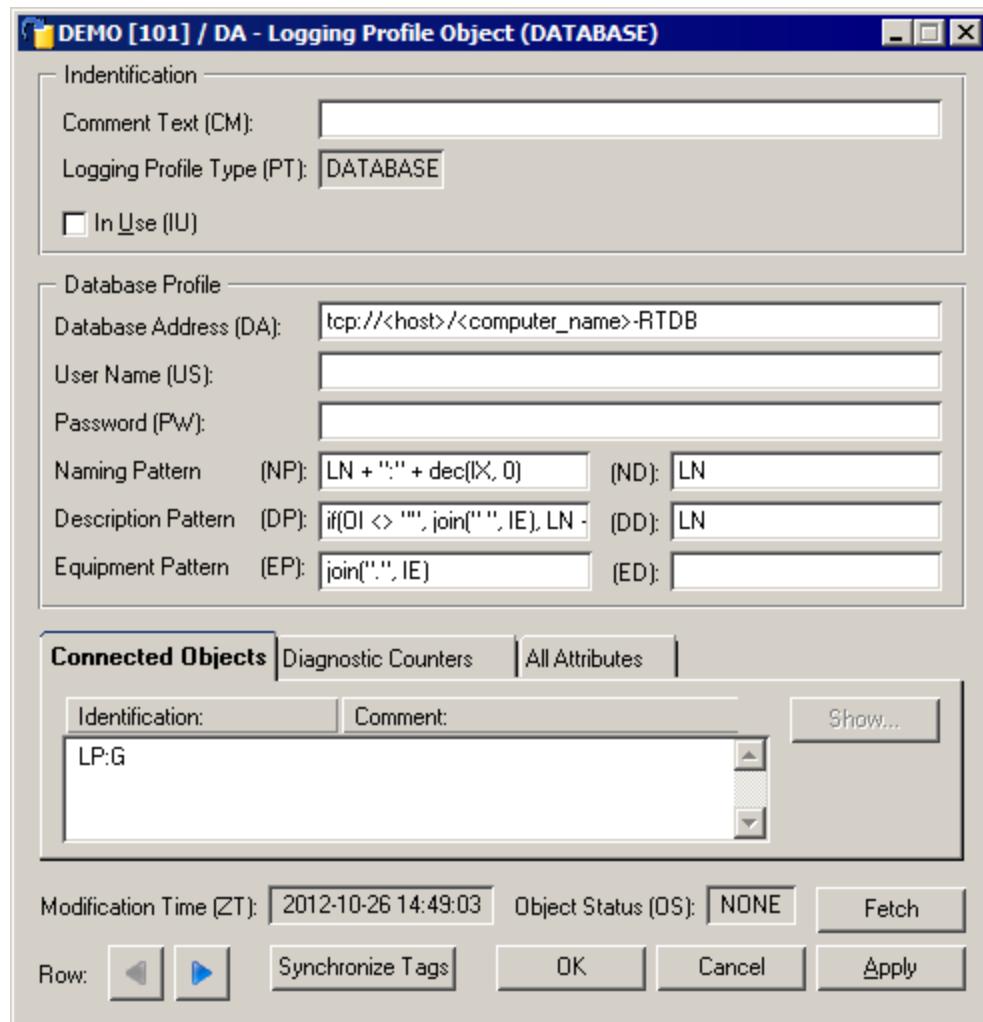


Figure 124: The Definition Tool of the Logging Profile Object of type DATABASE

24.5 Definition tool for logging profile object of type HISTORY

The definition tool of the **Logging Profile Object** of type HISTORY is shown in [Figure 125](#). See [Section 12.2](#) for the attributes.

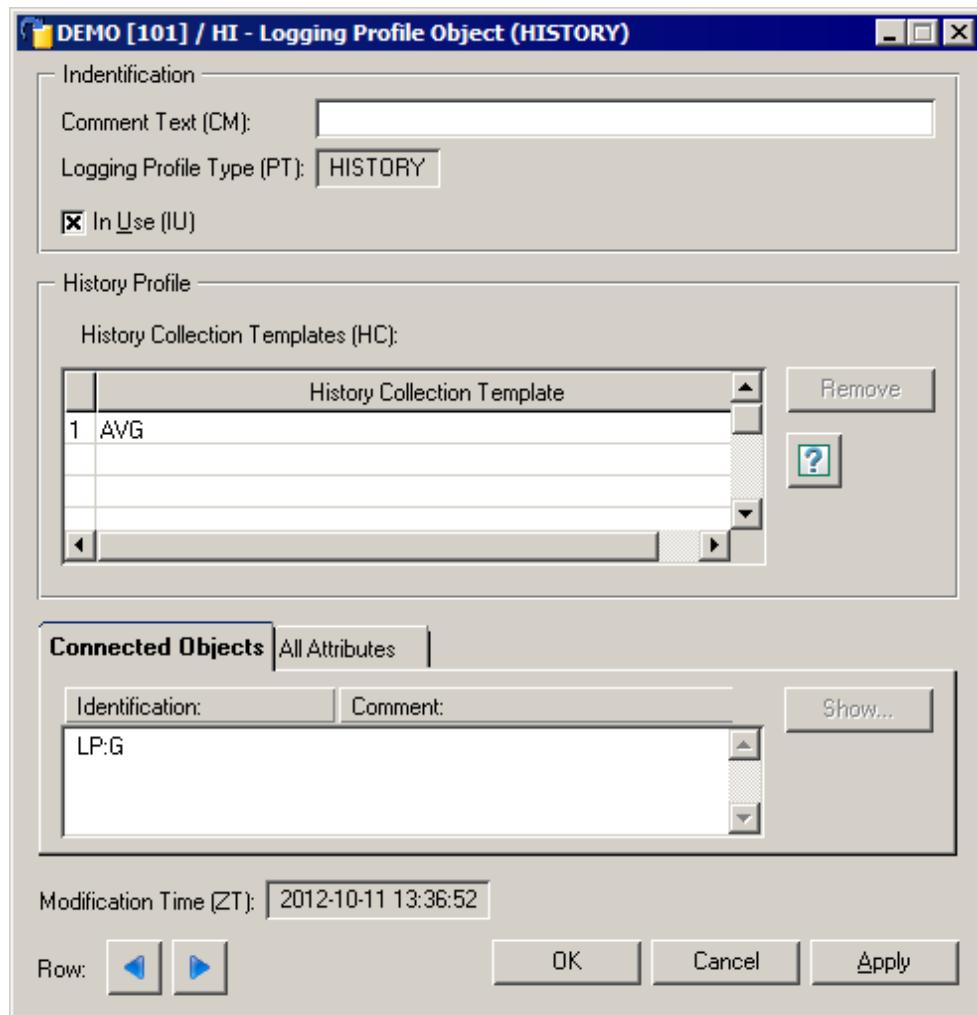


Figure 125: The Definition Tool of the Logging Profile Object of type HISTORY

Section 25 Free Type Object Definition Tools

25.1 General

This section introduces the Free Type Object Tool and the Free Type Process Object Tool. These tools are Application Object handling tools and are launched from the **Application Object Navigator**.

The tool for handling application objects, **Application Object Navigator**, is accessed through Tool Manager. The **Application Object Navigator** presents objects classified by type in a tree where application names form the nodes and object types form the leafs. The Free Type Object tools are accessed by choosing object type and name of the object of interest.

25.2 Free Type Process Object Tool

25.2.1 Accessing the tool

Clicking the Free Type Process Object type in the tree of **Application Object Navigator** shows the object names in the listbox to right. To access the Free Type Process Object tool, double-click an object name. The dialog box shown in [Figure 126](#) is opened.

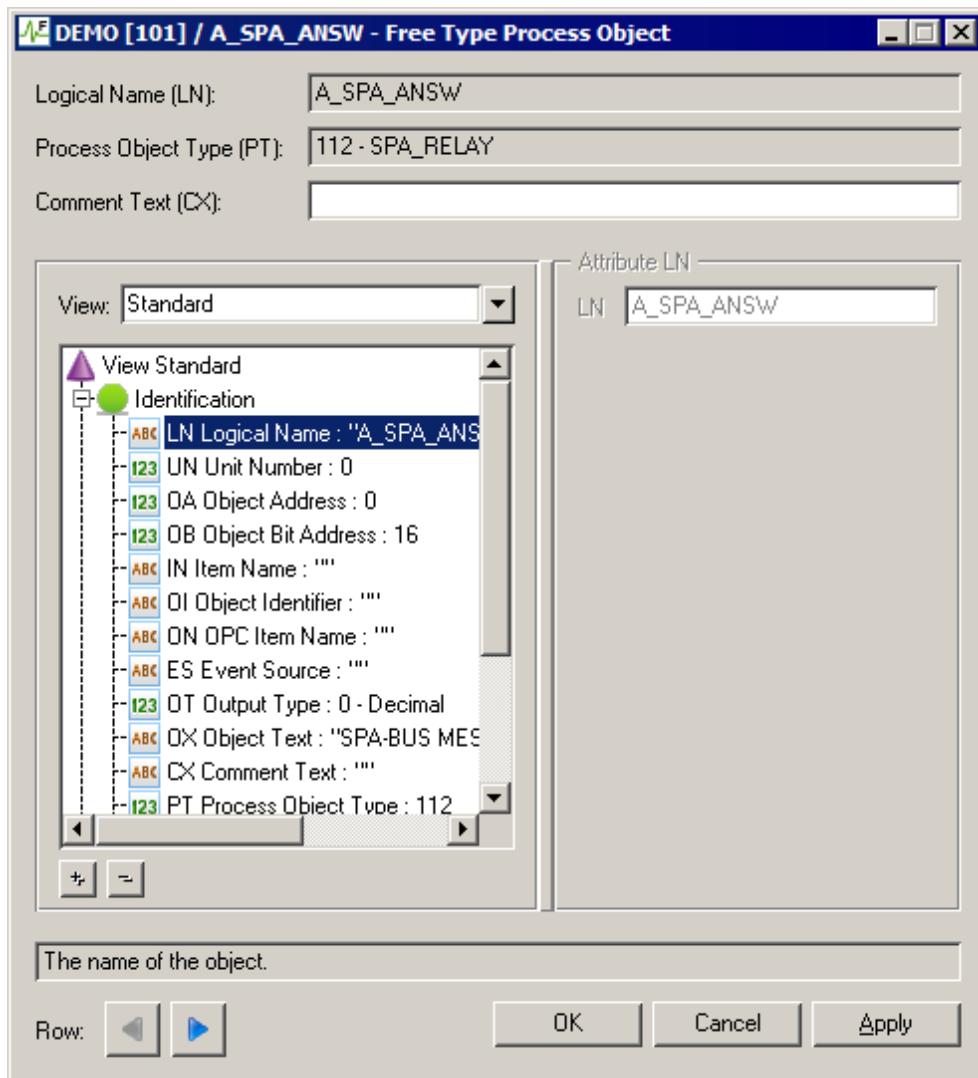


Figure 126: The Free Type Process Object tool as it appears after double clicking an object named 'A_SPA_ANSW' in the Object Navigator and expanding the 'Identification' node with the default view 'Standard'.

25.2.2 Fields

Logical Name (LN)	Value (text) of the LN-attribute. The logical name of the process object.
Process Object Type (PT)	Value (integer) of the PT-attribute.
Comment Text (CX)	Value (text) of the CX-attribute. The value of this attribute is supposed to serve as a description of the object.
View	This drop-down combo box offers two alternative views to the attribute tree. The alternative views are Standard and User Attributes.
Attribute tree	The attributes are displayed in a tree structure according to category.
Attribute box	Contains facilities for editing the attribute values displayed in the 'Attribute Tree'.
Attribute information	A description of the attribute or its function.

25.2.3 Using the tool

The Free Type Process Object tool is used for displaying and setting the values of attributes of free type process objects. For object identification the tool presents essential identification attributes (LN, PT and CX).

25.2.4 Storing settings and exiting the Tool

The value of some attributes shown in the attribute box may be edited, others only viewed.

- | | |
|---------------|--|
| OK | Save settings and close dialog. In case of an error the dialog stays open and the status code of the error is shown. |
| Cancel | Close the tool without saving. If settings have been changed during the session the user is asked to confirm the action. |
| Apply | Save settings and leave dialog open. Use for intermediate save. |

25.3 Free Type Object Tool

25.3.1 Accessing the tool

The Free Type Object tool is accessed from **Application Object Navigator** by double-clicking the **Free Type Objects** leaf in the tree.

25.3.2 Using the tool

The Free Type Object tool is used for displaying and setting the values of attributes of free type objects. For object identification, the tool presents essential identification attributes (LN and PT). The dialog consists of two main components: the **Attribute Definition** tab and the **Attribute Tree** tab.

25.3.3 Attribute definition

User attribute names are presented in a listbox. The order in which they are presented is determined by the elements in the text vector that is the value of the AN attribute. The value can be checked using the **Examine** tab of **Test Dialog** and entering OBJ_NAME:FAN in the **inspection** field.

The properties of a single User Attribute are shown in the right side of the **User Attributes** list. The properties displayed belong to the User Attribute currently selected in the **User Attributes** list. The values of AN, AI, AT and AL indexes cannot be configured on existing User Attributes.

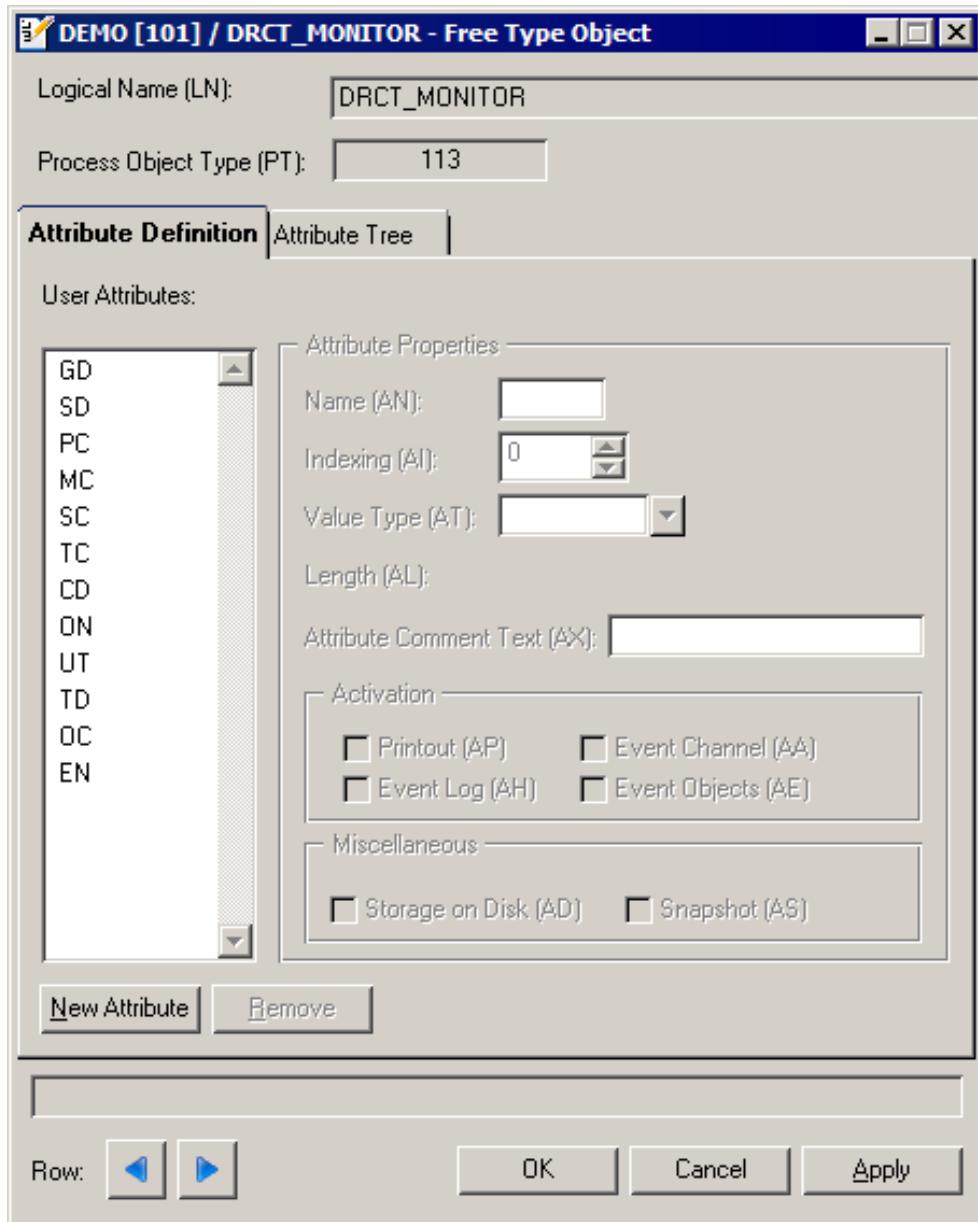


Figure 127: Attribute Definition tab of the Free Type Object definition tool

New User Attributes are defined by clicking the **New Attribute** button, see [Figure 127](#). This action adds a + sign to the attribute list, focuses the **Name (AN)** field and assigns default values to the rest of the attribute indices. As the User Attribute name is defined, the name is shown in the **User Attributes** list preceded by a + sign. The + sign disappears as the definition is saved. The attribute indices AN, AI, AT and AL are modifiable until the User Attribute definition is saved. Removal of User Attributes can be done prior to saving. Saved User Attributes cannot be removed. After saving the User Attributes, the attributes appear on the View User Attributes tree of the same type of process object in the Free Type Process Object Tool, see [Section 25.2](#).

25.3.4 Attribute tree

The **Attribute Tree** tab is used for grouping, displaying and modifying Free Type Object attributes. The groups and attributes displayed in the tree are predefined (in file attrib_f.scl). When an attribute in the tree is selected, an attribute data type specific edit box is displayed in the edit box.

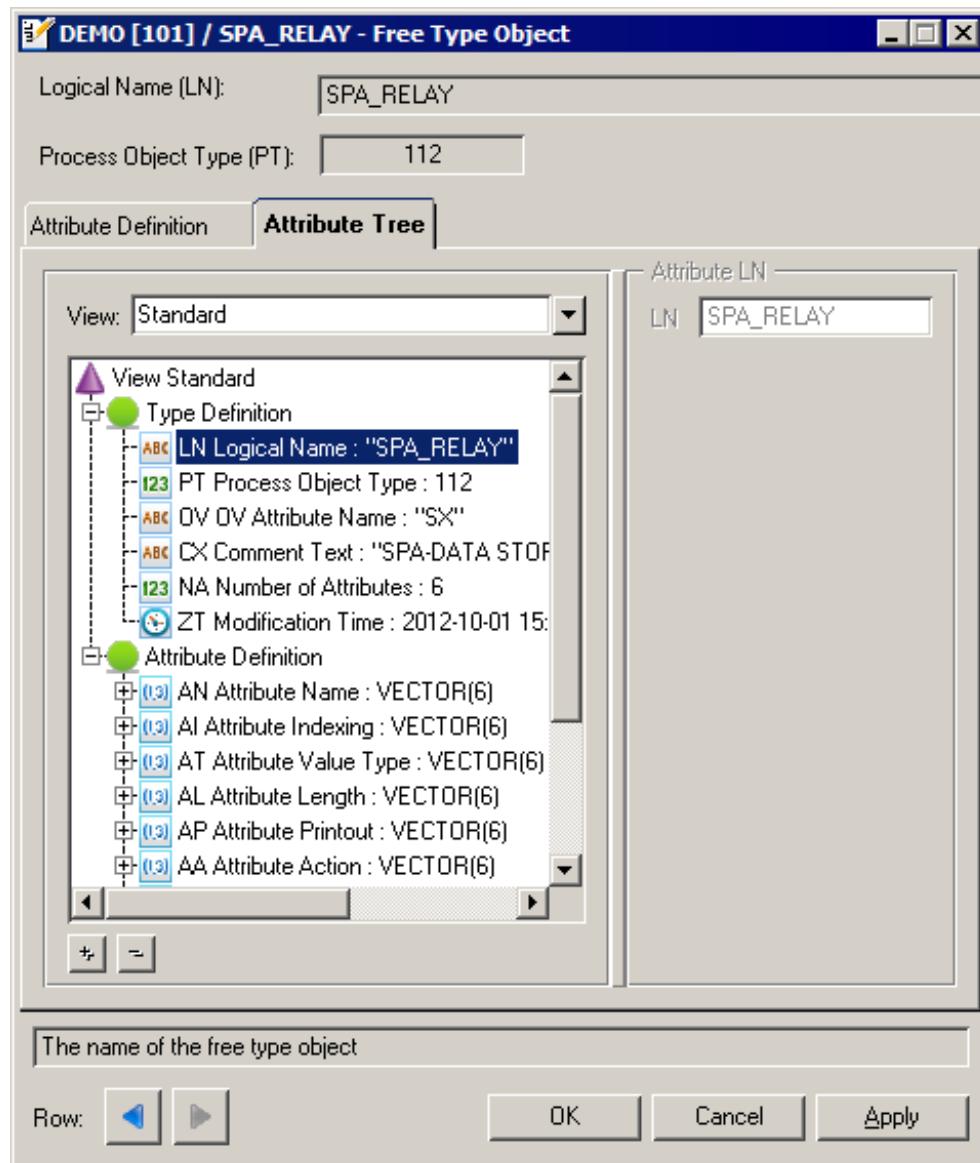


Figure 128: The Attribute Tree tab of Free Type Object definition tool

25.3.5 Storing settings and exiting the tool

The value of attribute indices only for viewing are dimmed.

- | | |
|---------------|---|
| OK | Save settings and close dialog. In case of an error the dialog stays open and the status code of the error is shown. |
| Cancel | Close the tool without saving. If settings have been changed during the session, the user is asked to confirm the action. |
| Apply | Save settings and leave dialog open. Use for intermediate save. |

Index

A	
AA :Action Activation (P).....	72
AA :Attribute Action (F).....	222
AB Alarm Blocking (P).....	98
AC Alarm Class (P).....	61
Acknowledgement.....	62, 63
Action Activation (P).....	72
Action at First Update (P).....	72
Action Enabled (P).....	72
Action Name (P).....	72
Action on History (P).....	72
Activation Blocking (P).....	100
AD :Alarm Delay (P).....	61
AD :Attribute on Disk (F).....	222
Addressing attributes.....	274
AE :Action Enabled (P).....	72
AE :Attribute Event (F).....	222
AEP_EVENT.....	193
AF Action at First Update (P).....	72
AG Alarm Generation (BI).....	59
AH :Action on History (P).....	72
AH :Attribute History (F).....	222
AI.....	45
AI :Analog Input (AI).....	87
AI :Attribute Indexing (F).....	220
AK OPC Ack Required (OE).....	112
AL :Alarm (P).....	94
AL :Attribute Length (F).....	221
Alarm.....	38
Alarm (P).....	94
Alarm Activation (DB).....	59
Alarm Activation (NT).....	60
Alarm blocking.....	56
Alarm Blocking (P).....	98
Alarm buffer.....	38
Alarm class.....	56
Alarm Class (P).....	61
Alarm delay.....	56
Alarm Delay (P).....	61
Alarm Generation (BI).....	59
Alarm limits.....	63
Alarm list.....	56, 225, 226
Alarm Milliseconds (P).....	95, 96
Alarm monitor.....	56, 62
Alarm On Qualified Time (P).....	98
Alarm On Time (P).....	98
Alarm On Time Milliseconds (P).....	97
Alarm picture.....	56, 62
Alarm Qualified Time (P).....	95, 97
Alarm Receipt (P).....	94
Alarms and Events Custom Interface Standard.....	108
Alarm Severity (P).....	95
Alarm State (P).....	94
Alarm Time (P).....	95, 97
Alarm Zone (AI).....	96
AM Alarm Milliseconds (P).....	95
AN :Action Name (P).....	72
AN :Attribute Name (F).....	220
Analog Input.....	38
Analog Input (AI).....	87
Analog Output.....	38
Analog Output (AO).....	87
AO.....	45
AO :Analog Output (AO).....	87
AO :Attribute Offset (F).....	223
AOR_EVENT.....	194
AP Attribute Printout (F).....	221
APL_ALARM.....	56, 190
APL_ALARM_ACK.....	190
APL_BLOCKING.....	190
APL_CLOSE.....	189
APL_EVENT.....	195, 198
APL_INIT_1.....	43, 189
APL_INIT_2.....	43, 189
APL_INIT_H.....	189
APL_REPORT... 152, 160, 166, 172, 175, 187, 206	
APL-APL diagnostics.....	196
Application.....	31
APPLICATION_ALARM_LIST.....	34
APPLICATION_DEFAULT (Daylight Switch Policy)....	178
APPLICATION_OBJECT_ATTRIBUTES.....	34
APPLICATION_OBJECT_LIST	34
Application Object Navigator.....	323
Application objects.....	24
AQ Alarm Qualified Time (P).....	95
AR Alarm Receipt (P).....	94
AS :Alarm State (P).....	94
AS :Attribute Snapshot (F).....	222
AT :Attribute Value Type (F).....	221
AT Alarm Time (P).....	95
Attribute.....	25, 40, 44
Attribute Access.....	33
Attribute access level.....	33
Attribute Action (F).....	222
Attribute box.....	324
Attribute Comment Text (F).....	223
Attribute Event (F).....	222
Attribute History (F).....	222
Attribute Indexing (F).....	220
Attribute indices.....	327
Attribute information.....	324
Attribute Length (F).....	221
Attribute name.....	30, 31, 32
Attribute Name (F).....	220
Attribute Offset (F).....	223
Attribute on Disk (F).....	222
Attribute Printout (F).....	221
Attribute Snapshot (F).....	222
Attribute tree.....	324, 326
Attribute Value Type (F).....	221
Audio alarm.....	57
Automatic printout.....	38, 82
AV : Alarm Severity (P).....	95
AX Attribute Comment Text (F).....	223
AZ Alarm Zone (AI).....	96
B	
BC Bit Count (PC).....	54
BI.....	45
BI Binary Input (BI).....	87
Binary Input.....	38
Binary Input (BI).....	87
Binary Output.....	38
Binary Output (BO).....	88
Bit Count (PC).....	54
Bit Stream.....	38

Bit Stream (BS).....	88
BL Blocked (P).....	103
Blocked (P).....	103
BO.....	45
BO Binary Output (BO).....	88
BOTH (OPC Event Generation).....	137
BS.....	45
BS Bit Stream (BS).....	88
C	
CA :Changed Attribute (Events).....	120
CA :Compression Accuracy (G).....	208
CALENDAR (Cycle Policy).....	177
CALENDAR (Daylight Switch Policy).....	178
Cause of Transmission (P).....	103
CC Control Supervision Configuration (Group s).....	124
CD :Condition (T).....	176
CD :Event Category Description (H).....	138
CD Configuration Data (Groups).....	125
CE Counter Enabled (BI, BO, DB, OE).....	84
Changed Attribute (Events).....	120
CI Event Category ID (H).....	138
CK OPC Cookie (OE).....	112
CL Counter Limit (BI, BO, DB, OE).....	85
CM :Comment (A).....	187
CM :Comment (C).....	167
CM :Comment (D).....	153
CM :Comment (G).....	207
CM :Comment (T).....	175
CM :OPC Change Mask (OE).....	112
CN :Condition Name (H).....	139
CN :Connected Network Objects (NT).....	83
CN :OPC Condition Name (OE).....	112
CO Counter Overflow (BI, BO, DB, OE).....	100
Command Procedure.....	305
Command procedure object.....	24
Command procedures.....	165, 305, 309
Command Qualifier (P).....	107
Comment :A.....	187
Comment :C.....	167
Comment :D.....	153
Comment :G.....	207
Comment :T.....	175
Comment Text.....	324
Comment Text :F.....	219
Comment Text :H.....	132
Comment Text :P.....	46
Compilation State (C).....	169
Compiled Program (C).....	169
Compile IN when Edited.....	305
Compression Accuracy (G).....	208
CONDITION.....	137, 173
CONDITION (Event Type).....	139
Condition (T).....	176
Condition Name (H).....	139
Configuration Data.....	0 , 6
Connected Network Objects (NT).....	83
Connectivity Packages.....	29
Control Status (BO, DO, AO).....	92
Control Supervision.....	56
Control Supervision Configuration (Groups)...	124
COPY (Logging Function).....	156
Counter Enabled (BI, BO, DB, OE).....	84
Counter Limit (BI, BO, DB, OE).....	85
Counter Overflow (BI, BO, DB, OE).....	100
Counter Value (BI, BO, DB, OE).....	100
CP :Compiled Program (C).....	169
CP :Cycle Policy (T).....	177
CQ OPC Cookie Time (OE).....	112
CREATE.....	34
Creating objects.....	34
CS :Compilation State (C).....	169
CS :Control Status (BO, DO, AO).....	92
CT Cause of Transmission (P).....	103
CV Counter Value (BI, BO, DB, OE).....	100
CX :Comment Text (F).....	219
CX :Comment Text (H).....	132
CX :Comment Text (P).....	46
CY.....	310
Cycle.....	173
Cycle (T).....	177
Cycle Policy (T).....	177
CY Cycle (T).....	177
D	
DA Database Address (G).....	209
Data.....	149
Database.....	26
Database Address (G).....	209
Datalog object.....	149
Data object.....	24, 149, 297
Data type.....	32
Daylight Switch Policy (T).....	177
DB.....	45
DB Double Binary Indication (DB).....	88
DC Diagnostic Counters (G).....	209
DC Directory Contents (FT).....	116
DD Description Pattern D (G).....	210
Decimal Places (AI, AO).....	54
Defining objects.....	26
Definition tools.....	305
DELETE.....	35
Deleting objects.....	35
Description Pattern D (G).....	210
Description Pattern P (G).....	210
DI.....	45
DI :Discreteness (G).....	208
Diagnostic Counters (G).....	209
DI Digital Input (DI).....	89
DIFFERENCE (Logging Function).....	156
Digital Input.....	38
Digital Input (DI).....	89
Digital Output.....	38
Digital Output (DO).....	89
DIRECT (Logging Function).....	156
Directive Text (P).....	85
Directory Contents (FT).....	116
Discreteness (G).....	208
DO.....	45, 165
DO Digital Output (DO).....	89
Double Binary Indication (DB).....	88
Double Indication.....	38
DP :Daylight Switch Policy (T).....	177
DP :Decimal Places (AI, AO).....	54
DP Description Pattern P (G).....	210
DX Directive Text (P).....	85
E	
ED Equipment Path Pattern D (G).....	210
ED Event Daylight Saving (Events).....	120
EE Event Enabled (P).....	73
EH.....	129
EH Event Handling (P).....	67
EM :Event Messages (H).....	139
EM :Event Time Milliseconds (Events).....	121
End of Period (PC).....	105
Engineering Unit (AI, AO, PC).....	55
EP :End of Period (PC).....	105

EP :Execution Priority (C).....	171
EP :Execution Priority (D).....	159
EQ Qualified Event Time (Events).....	121
Equipment Path Pattern D (G).....	210
ES Event Source (P).....	46
ET :Event Time (Events).....	121
ET :Event Type (H).....	139
EVEN (Cycle Policy).....	177
EVEN (Daylight Switch Policy).....	178
Event Category.....	79
Event Category Description (H).....	138
Event Category ID (H).....	138
Event channel.....	38, 72, 185
Event channel object.....	24
Event channel queue.....	151
Event channels.....	315
Event Comment Text (Events).....	121
Event Daylight Saving (Events).....	120
Event Enabled (P).....	73
Event Handling.....	129, 287
Event Handling (P).....	67
Event handling object.....	24
Event Handling Type (H).....	132
Event list.....	225
Event Messages (H).....	139
Event method.....	213
Event object.....	24, 34, 38, 213
Event object handling.....	34
Event recording object.....	44
Event recording objects.....	37
Event Source.....	79
Event Source (P).....	46
Event Time (Events).....	121
Event Time Milliseconds (Events).....	121
Event Type (H).....	139
EXEC.....	34, 150, 165, 186, 214, 215
EXEC_AFTER.....	150, 165, 186, 214, 215
Executing objects.....	34
Executing task.....	152
Execution.....	169, 173
Execution Control page.....	306
Execution Priority :C.....	171
Execution Priority :D.....	159
Execution queue.....	151
EX Event Comment Text (Events).....	121
F	
Fetch.....	273
FF File Function (FT).....	116
FI Free Integer :C.....	167
FI Free Integer :D.....	154
FI Free Integer :P.....	85
File Function (FT).....	116
File Name (FT).....	116
File Transfer.....	38
File Transfer (FT).....	117
File Transfer Progress (FT).....	117
FN File Name (FT).....	116
FP File Transfer Progress (FT).....	117
Free Integer :C.....	167
Free Integer :D.....	154
Free Integer :P.....	85
Free Text :C.....	167
Free Text :D.....	154
Free Text :P.....	85
Free type object.....	24
Free type objects.....	217
Free Type Object Tool.....	323
Free Type Process Object Tool.....	323
FT.....	45
FT File Transfer (FT).....	117
FX Free Text :C.....	167
FX Free Text :D.....	154
FX Free Text :P.....	85
G	
GA Group Alarm (Groups).....	125
Gateway Information (P).....	85
GB Group Blockings (Groups).....	125
GC Group Comment (Groups).....	126
GET.....	34
GI Gateway Information (P).....	85
GN Logging Name :D.....	158
GN Logging Name :P.....	77
GP Logging Profile :D.....	159
GP Logging Profile :P.....	77
Group.....	43, 44
Group Alarm (Groups).....	125
Group Alarm State (Groups).....	126
Group Blockings (Groups).....	125
Group Comment (Groups).....	126
Grouping.....	326
Group Type (Groups).....	126
GS Group Alarm State (Groups).....	126
GT Group Type (Groups).....	126
H	
HA History Activation (P).....	76
HB History Blocking (P).....	99
HC History Collection Templates (G).....	212
HD History Logging Daylight Saving (Events).....	121
HE History Enabled (P).....	76
HF History at First Update (P).....	76
HH History on History (P).....	76
High Input (AI).....	64
High Output (AO).....	64
High Warning (AI).....	64
HI High Input (AI).....	64
Historian.....	77, 158
HISTORIAN_EVENT.....	193
Historical data.....	149
HISTORY_DATABASE_MANAGER.....	34, 75
History Activation (P).....	76
History at First Update (P).....	76
History Blocking (P).....	99
History buffering.....	75
History Collection Templates (G).....	212
History database.....	38, 75
History Enabled (P).....	76
History File Number :C.....	172
History File Number :D.....	160
History Logging Daylight Saving (Events).....	121
History Logging Time (Events).....	122
History Logging Time Milliseconds (Events)....	122
History Log Numbers (P).....	77
History on History (P).....	76
History Registrations (D).....	155
HL History Log Numbers (P).....	77
HM History Logging Time Milliseconds (Even ts)....	122
HN :History File Number (C).....	172
HN :History File Number (D).....	160
HO High Output (AO).....	64
HQ Qualified History Logging Time (Events)....	122
HR History Registrations (D).....	155
HT :Event Handling Type (H).....	132
HT :History Logging Time (Events).....	122

HW High Warning (AI).....	64
I	
ID :Identification (FT).....	117
ID :OPC Actor ID (OE).....	113
Identification (FT).....	117
Identifier Elements (P).....	46
Identifier List (P).....	47
IEC.....	107
IE Identifier Elements (P).....	46
IG Item Group (P).....	49
IL Identifier List (P).....	47
IN :Instruction (D).....	155
IN :Instructions (C).....	169
IN :Item Name (P).....	49
Index.....	32
Index (P).....	44
Initialisation.....	150, 173
Instruction (D).....	155
Instructions (C).....	169
Integer Representation (AI, AO).....	54
INTEGRAL (Logging Function).....	156
In Use :C.....	168
In Use :D.....	154
In Use :G.....	207
In Use :P.....	52
In Use :T.....	176
INVALID_OPC_ITEM.....	191
IP_EVENT.....	197
IR Integer Representation (AI, AO).....	54
Item Group (P).....	49
Item Name (P).....	49
IU :In Use (C).....	168
IU :In Use (D).....	154
IU :In Use (G).....	207
IU :In Use (P).....	52
IU :In Use (T).....	176
IX Index (P).....	44
K	
KM Acknowledgement Milliseconds (P).....	96
KQ Acknowledgement Qualified Time (P)....	97
KT Acknowledgement Time (P).....	97
L	
LA Alarm Activation (DB).....	59
LA Alarm Activation (NT).....	60
Latest Registration (D).....	161
LD Listing Device (P).....	81
LE Level Enumeration (NT).....	118
Level Enumeration (NT).....	118
Level Text (NT).....	119
Level Value (NT).....	119
LF :Logging Function (D).....	156
LF :Logical Format (Groups).....	127
LIB500.....	29
LI Low Input (AI).....	64
Linear scaling.....	145
LIST.....	127, 225
Listing Device (P).....	81
LN Logical Name :A.....	187
LN Logical Name :C.....	168
LN Logical Name :D.....	154
LN Logical Name :F.....	219
LN Logical Name :G.....	207
LN Logical Name :Groups.....	127
LN Logical Name :H.....	132
LN Logical Name :P.....	45
LN Logical Name :T.....	176
LN Logical Name :X.....	146

Logging function.....	149
Logging Function (D).....	156
Logging Name :D.....	158
Logging Name :P.....	77
Logging Profile :D.....	159
Logging Profile :P.....	77
Logging profile object.....	24
Logical Format (Groups).....	127
Logical Name.....	324
Logical Name :A.....	187
Logical Name :C.....	168
Logical Name :D.....	154
Logical Name :F.....	219
Logical Name :G.....	207
Logical Name :Groups.....	127
Logical Name :H.....	132
Logical Name :P.....	45
Logical Name :T.....	176
Logical Name :X.....	146
LO Low Output (AO).....	65
Loop State (NT).....	118
Low Input (AI).....	64
Low Output (AO).....	65
Low Warning (AI).....	65
LP Loop State (NT).....	118
LR Latest Registration (D).....	161
LV Level Value (NT).....	119
LW Low Warning (AI).....	65
LX Level Text (NT).....	119
M	
MAXIMUM (Logging Function).....	156
Maximum Qualified Time (AI).....	102
Maximum Time (AI).....	102
Maximum Time Milliseconds (AI).....	102
Maximum Value (AI).....	102
MEAN (Logging Function).....	156
Memory Only :C.....	172
Memory Only :D.....	160
Memory Only :T.....	176
Message Text (Events).....	122
Message Texts (H).....	132
MINIMUM (Logging Function).....	156
Minimum Qualified Time (AI).....	101
Minimum Time (AI).....	101
Minimum Time Milliseconds (AI).....	101
Minimum Value (AI).....	101
MIRRORING_CONFIGURATION.....	192
MM Minimum Time Milliseconds (AI).....	101
Modification Time :A.....	189
Modification Time :C.....	168
Modification Time :D.....	155
Modification Time :F.....	220
Modification Time :G.....	207
Modification Time :Groups.....	127
Modification Time :H.....	136
Modification Time :P.....	45
Modification Time :T.....	176
Modification Time :X.....	147
MODIFY.....	35
MO Memory Only :C.....	172
MO Memory Only :D.....	160
MO Memory Only :T.....	176
MON_EVENT.....	191
MQ Minimum Qualified Time (AI).....	101
MT :Message Texts (H).....	132
MT :Minimum Time (AI).....	101
MV Minimum Value (AI).....	101
MX :Message Text (Events).....	122
MX :Translated Message Texts (H).....	133

N

Name Pattern D (G).....	211
NA Number of Attributes (F).....	220
ND :Network Topology Data (NT).....	119
ND :Node (H).....	140
ND Name Pattern D (G).....	211
Network Feeds (NT).....	120
Network Object Connection (P).....	83
Network Object State (NT).....	89
Network Object Subtype (NT).....	83
Network Topology Data (NT).....	119
Network Topology Model (NT).....	83
Next Group.....	273
Next Index.....	273
NF Network Feeds (NT).....	120
NM Network Topology Model (NT).....	83
NO (Start-up Execution Policy).....	178
Node (H).....	140
NO Network Object Connection (P).....	83
Normal Value (Bl, DB).....	60
NP Name Pattern P (G).....	211
NS :Network Object Subtype (NT).....	83
NS :OPC New State (OE).....	113
NT Network Object State (NT).....	89
Number of Attributes (F).....	220
NV Normal Value (Bl, DB).....	60

O

OA Object Address (P).....	50
Object Address (P).....	50
Object attribute notation.....	30
Object Bit Address (P).....	51
Object identification.....	325
Object Identifier (P).....	47
Object name.....	31
Object Name (A).....	188
Object Navigator.....	305, 309
Object notation.....	30, 33
Object Status :C.....	169
Object Status :D.....	161
Object Status :G.....	211
Object Status :P.....	92
Object Text (P).....	48
Object Type (A).....	188
Object Value.....	87, 149
Object Value :D.....	161
Object Value :P.....	90
OB Object Bit Address (P).....	51
OC OPC Condition Event (H).....	136
OE.....	45, 108
OE OPC Event (OE).....	90, 113
OF Overflow (P).....	105
OG :OPC Event Generation (H).....	137
OG :Originator Identification (P).....	107
OI Object Identifier (P).....	47
ON.....	34, 213, 215
ON :Object Name (A).....	188
ON :OPC Item Name (C).....	168
ON :OPC Item Name (D).....	154
ON :OPC Item Name (P).....	48
ONCE (Start-up Execution Policy).....	179
One-to-one scaling.....	145
OPC.....	49
OPC A&E Server.....	108
OPC Ack Required (OE).....	112
OPC Active Time (OE).....	113
OPC Actor ID (OE).....	113
OPC Alarms & Events.....	46, 78
OPC Alarms and Events.....	38

OPC Alarms and Events Client.....	129
OPC Alarms and Events Server.....	129
OPC Change Mask (OE).....	112
OPC Condition Event (H).....	136
OPC Condition Name (OE).....	112
OPC Cookie (OE).....	112
OPC Cookie Time (OE).....	112
OPC Data Access Server.....	88
OPC Event (OE).....	90, 113
OPC Event Category (OE).....	114
OPC Event Generation (H).....	137
OPC Event Message (OE).....	114
OPC Event Time (OE).....	115
OPC Event Type (OE).....	115
OPC Foundation.....	108
OPC Item Name :C.....	168
OPC Item Name :D.....	154
OPC Item Name :P.....	48
OPC New State (OE).....	113
OPC Quality (OE).....	114
OPC Severity (OE).....	114
OPC Simple Event (H).....	138
OPC Subcondition Name (OE).....	114
Operating System Event Handler.....	202
OQ OPC Active Time (OE).....	113
Originator Identification (P).....	107
OR Out of Range (P).....	104
OS.....	38, 59
OS_EVENT.....	202
OS :Object Status (C).....	169
OS :Object Status (D).....	161
OS :Object Status (G).....	211
OS :Object Status (P).....	92
OS :OPC Simple Event (H).....	138
OSEH.....	202
OT :Object Type (A).....	188
OT :Output Type (P).....	51
Out of Range (P).....	104
Output Type (P).....	51
OV.....	38
OV :Object Value (D).....	161
OV :OV Attribute Name (F).....	219
OV Attribute Name (F).....	219
Overflow (P).....	105
OV Object Value :P.....	90
OX Object Text (P).....	48
P	
PA Printout Activation (P).....	81
Parallel Execution :C.....	171
Parallel Execution :D.....	159
Parallel Execution :T.....	180
Parallel Queue :C.....	171
Parallel Queue :D.....	159
Parallel Queue :T.....	180
Parallel queues.....	151
Password (G).....	212
PB Printout Blocking (P).....	99
PC.....	45
PC Pulse Counter (PC).....	91
PD Picture Devices (P).....	62
PE Parallel Execution :C.....	171
PE Parallel Execution :D.....	159
PE Parallel Execution :T.....	180
PF Physical Format (P).....	82
PH Printout on History (P).....	82
Physical Format (P).....	82
Picture (P).....	62
Picture Devices (P).....	62
PI Picture (P).....	62

PO Post-processing Based on Object Value (P)	68
Post-processing Based on Object Value (P)	68
PQ Parallel Queue :C	171
PQ Parallel Queue :D	159
PQ Parallel Queue :T	180
Previous Group	273
Previous Index	273
Printer	81
Printout	81
Printout Activation (P)	81
Printout at First Update (P)	82
Printout Blocking (P)	99
Printout on History (P)	82
Priority	173
Process database	26, 37, 38, 43, 218
Process object	24, 214
Process object group	42
Process object notation	43
Process objects	37, 38, 45
Process Object Type	324
Process Object Type :F	219
Process Object Type :P	45
Process Views (Groups)	127
PROD_QUERY	75
Profile Type (G)	207
PS Pulse Scale (D)	157
PT :Process Object Type (F)	219
PT :Process Object Type (P)	45
PT :Profile Type (G)	207
Pulse Counter	38
Pulse Counter (PC)	91
PULSE DERIVATIVE (Logging Function)	156
PULSE DIFFERENCE (Logging Function)	156
Pulse Scale (D)	157
PU Printout at First Update (P)	82
PV :Previous Value (Events)	122
PV :Process Views (Groups)	127
PW Password (G)	212
Q	
QB Qualified Begin Time (T)	181
QE Qualified End Time (T)	182
QL Command Qualifier (P)	107
QS Qualified Synchronisation Time (T)	182
QT Qualified Registration Time :C	170
QT Qualified Registration Time :D	162
QT Qualified Registration Time :T	182
Qualified Begin Time (T)	181
Qualified End Time (T)	182
Qualified Event Time (Events)	121
Qualified History Logging Time (Events)	122
Qualified Registration Time :C	170
Qualified Registration Time :D	162
Qualified Registration Time :T	182
Qualified Synchronisation Time (T)	182
QU OPC Quality (OE)	114
R	
RA Reserved A (P)	104
Raw value	149
RB :Registered Begin Time (T)	182
RB :Reserved B (P)	104
RC Receipt (P)	62
Reading an attribute	32
Receipt (P)	62
Referenced Objects (NT)	84
Registered Begin Time (T)	182
Registered End Time (T)	183
Registered Synchronisation Time (T)	183
Registration Milliseconds (P)	93
Registration Qualified Time (P)	93
Registration Time :C	170
Registration Time :D	162
Registration Time :P	93
Registration Time :T	183
Report database	26, 152, 166, 175, 185, 187, 206
Report object	26
RE Registered End Time (T)	183
Reserved A (P)	104
Reserved B (P)	104
Reserved Integer (P)	85
Reserved Text (P)	86
Reserved Z (P)	86
RI Reserved Integer (P)	85
RM	38
RM Registration Milliseconds (P)	93
RO Referenced Objects (NT)	84
RQ	38
RQ Registration Qualified Time (P)	93
RS Registered Synchronisation Time (T)	183
RT Registration Time :C	170
RT Registration Time :D	162
RT Registration Time :P	93
RT Registration Time :T	183
RTU	38, 50
RX Reserved Text (P)	86
RZ Reserved Z (P)	86
S	
SA Scaling Algorithm (X)	146
SB Substituted (SB)	105
SC :Scaling (PC)	55
SC :Scaling Constants (X)	146
SCADA Zone Supervision (AI)	65
Scale Name (AI, AO)	55
Scale object	24
Scales	295
Scaling (PC)	55
Scaling algorithm	145
Scaling Algorithm (X)	146
Scaling Constants (X)	146
SCIL	26, 29, 30, 33
SCIL commands	33
SCIL program	165
SCL files	30
SE :OPC Severity (OE)	114
SE :Selection (AO, BO)	106
SE :Start-up Execution (C)	171
SE :Start-up Execution (D)	160
Secondary Object Names (A)	188
Secondary Object Types (A)	188
Selection (AO, BO)	106
SET	32, 34
Setting attribute values	34
SIMPLE_OR_TRACKING (OPC Event Generation)	n) 137
SIMPLE (Event Type)	139
SN	145
SN :OPC Subcondition Name (OE)	114
SN :Scale Name (AI, AO)	55
SN :Secondary Object Names (A)	188
SN :Subcondition Names (H)	140
Snapshot variables	70
Source (D)	157
SP :Start-up Execution Policy (T)	178
SP :Stop Execution (BO)	106
SR Source (D)	157
SS	37

SS Switch State (P).....	53
ST :Engineering Unit (AI, AO, PC).....	55
ST :Secondary Object Types (A).....	188
ST :State Texts (H).....	134
ST :Status (FT).....	117
ST :Storage (G).....	208
Start-up Execution :C.....	171
Start-up Execution :D.....	160
Start-up Execution Policy (T).....	178
State Text (P).....	91
State Texts (H).....	134
Status (FT).....	117
Stepwise linear scaling.....	145
Stop Execution (BO).....	106
Storage.....	43
Storage (G).....	208
String based addressing.....	50
SU :Substitution State (P).....	53
SU :Synchronisation Unit (T).....	179
Subcondition Names (H).....	140
Substituted (SB).....	105
Substitution State (P).....	53
SUM (Logging Function).....	156
Switch State (P).....	53
SX :State Text (P).....	91
SX :Synchronised Execution (T).....	180
SX :Translated State Texts (H).....	134
Synchronisation.....	173, 179
Synchronisation Time (T).....	179
Synchronisation Unit (T).....	179
Synchronised Execution (T).....	180
SYS_TEXT.SDB.....	141
SY Synchronisation Time (T).....	179
SZ SCADA Zone Supervision (AI).....	65

T

Table Index (P).....	52
TC Time Channel :C.....	172
TC Time Channel :D.....	160
Test Mode (TM).....	105
Threshold (AI).....	73
TH Threshold (AI).....	73
Time channel.....	173, 309
Time Channel :C.....	172
Time Channel :D.....	160
Time channel object.....	24
Time channel queue.....	151
TIME DERIVATIVE (Logging Function).....	156
Time stamp.....	38, 93
Time Stamp :C.....	170
Time Stamp :D.....	157
TI Table Index (P).....	52
TM Test Mode (TM).....	105
Topological State (BI, DB).....	91
TRACKING (Event Type).....	139
Translated Message Texts (H).....	133
Translated Object Text (P).....	48
Translated State Texts (H).....	134
Trend data.....	149
TS :Time Stamp (C).....	170
TS :Time Stamp (D).....	157
TS :Topological State (BI, DB).....	91
TX Translated Object Text (P).....	48
Type.....	31
Type Identification (P).....	108
TY Type Identification (P).....	108

U

UAL_EVENT.....	194
UB Update Blocking (P).....	99

UNDEF_OPC_EVENT.....	191
UNDEF_PROC.....	191
Unit Number (P).....	52
UN Unit Number (P).....	52
Update Blocking (P).....	99
Updating process database.....	34
User Attribute list.....	325
User Attributes.....	325
User-defined attribute.....	217, 220
User Name :Events.....	123
User Name :G.....	212
US User Name :Events.....	123
US User Name :G.....	212

V

Validation stamp.....	38
Value Count (H).....	134
Value Formula (H).....	135
Value Length (D).....	158
Value Low (H).....	136
Value Type (D).....	158
Variable.....	150, 225
Variable object.....	24, 225
Variables.....	166
VC :OPC Event Category (OE).....	114
VC :Value Count (H).....	134
VF Value Formula (H).....	135
View.....	324
VL :Value Length (D).....	158
VL :Value Low (H).....	136
VM OPC Event Message (OE).....	114
VQ OPC Event Time (OE).....	115
VT_BSTR.....	88
VT :OPC Event Type (OE).....	115
VT :Value Type (D).....	158

W

Warning Alarm Class (P).....	63
Warning Alarm Receipt (P).....	63
Warning limits.....	63
Warning Qualified Time (AI).....	97
WC Warning Alarm Class.....	63
WQ Warning Qualified Time (AI).....	97
Writing an attribute.....	32
WR Warning Alarm Receipt (P).....	63

X

XB Activation Blocking (P).....	100
XM Maximum Time Milliseconds (AI).....	102
XQ Maximum Qualified Time (AI).....	102
XT Maximum Time (AI).....	102
XV Maximum Value (AI).....	102

Y

YES (Start-up Execution Policy).....	178
YM Alarm On Time Milliseconds (P).....	97
YQ Alarm On Qualified Time (P).....	98
YT Alarm On Time (P).....	98

Z

ZD Zero Deadband (AI).....	66
Zero deadband.....	64
Zero Deadband (AI).....	66
Zero Deadband Supervision Enabled (AI)....	65
ZE Zero Deadband Supervision Enabled (AI)....	65
ZT Modification Time :A.....	189
ZT Modification Time :C.....	168
ZT Modification Time :D.....	155
ZT Modification Time :F.....	220
ZT Modification Time :G.....	207

ZT Modification Time :Groups.....	127
ZT Modification Time :H.....	136
ZT Modification Time :P.....	45
ZT Modification Time :T.....	176
ZT Modification Time :X.....	147

Hitachi ABB Power Grids
Grid Automation Products
PL 688
65101 Vaasa, Finland



Scan this QR code to visit our website

<https://hitachiabb-powergrids.com/microscadax>