2-Internal Channel

Internal Channel

The Internal Channel configuration defines virtual field units that have several special internal purposes, including an ISaGRAF Field Unit, Status Field Unit, Virtual RTU, Segment field unit, or Internal Master. The Internal Channel configuration and its 'child' objects are similar in structure to the Master Channel and its 'child' objects, but they refer to internal databases rather than external devices being polled by a communication protocol.

Attributes	Function
Object Type	Internal Channel
Parent(s)	System Clients Master Channels
Instance	0 to 15 (typically 15 is reserved for the Internal Channel, but instance 14 might also be used if a second ISaGRAF RTU is included) The Internal Channel must have at least Circuit child object defined under it. The Database Flush (DumpRTDB) object is an optional child under Internal Master.

Properties	Values
Name	Enter the Internal Channel name. *This is the name which appears in the MMI when viewing diagnostics.*
Channel Type	*In some configurations this may be listed as "Internal Channel", which includes a few operational differences noted below.* *The main differences between the Channel Types are listed below:* ***Internal Channel*** • Global Scan Period • One failed poll changes RTU comms status. ***Internal Channel Flex Scan*** • Scan Period configured for each scan. • All "effective" polls must fail before RTU comms status fails.
Auto Start	Select from the drop down menu the automatic polling required. *Automatic polling types supported are:* • *Yes – polling started automatically upon power-up* • *No – polling started manually through the MMI*
Response Timeout	Enter the response timeout in milliseconds. *This should normally be 32767 for the Internal Channel.*
Broadcast Delay	Enter the broadcast delay in milliseconds. *Delay in milliseconds after a command is sent to the ISaGRAF RTU, before normal polling resumes.*
Interpoll Delay	Enter the interpoll delay in milliseconds. Use this to add a delay between each poll sent by the channel to the ISaGRAF unit. *Time in milliseconds to wait between each poll.*

Scan Effective Limit	The Scan Effective Limit is the time (in seconds) defining which scans in the Scan Table are considered "effective" – meaning, polls which affect the status of the field unit if there are poll failures. Scan Table entries which have a Scan Period greater than the Scan Effective Limit do not mark the Field Unit offline when the scan fails. *For instance, if the Scan Effective Limit is configured for 30 seconds, then any scans defined with Scan Period <= 30 will be used to mark the Field unit online or offline. Scans with Scan Period greater than 30 will not mark the Field Unit offline even if they fail. The Scan Effective Limit only applies to the "Internal Channel Flex Scan Table" version of the channel object.*
Network Recovery	Enter the network recovery period in seconds. Time period to wait after an RTU fails, before attempting to re-establish communications with that RTU.
Scan Table	Click the **Edit Table** button to define the order and selection of polls to be sent to the ISaGRAF Field unit on this channel. Scan Table details: **Unit Address** - Enter the Modbus address of the ISaGRAF field unit to be polled. (To force the Scan Table to ignore the Scan Period, enter a Scan Table row with the unit address of -1.) **Poll Record** - This is the row number in the Poll Record for each poll defined for the ISaGRAF unit. The first row in a Poll Table is referenced as record 1. Only those polls which are to be scanned continuously need to be listed in this Scan Table. **Scan Period** - Enter the scan period in seconds. The Scan Period is the amount of time to use for scheduling each scan (global for all scans in the Internal Channel, or configured per scan in the Internal Channel Flex Scan Table). For the standard Internal Channel, the channel will restart the scan table sequence after the Scan Period has expired. If the total time for a given channel exceeds the scan period, the next scan shall be scheduled immediately. For the Internal Channel Flex Scan Table, each poll is scheduled based on its own Scan Period. If the total time required for scans at any point is greater than allowed by the Scan Periods, the scans will operate as fast as possible. Setting a Scan Period to -1 will disable a scan. **Comment** - Optional column, allowing a descriptive comment to be entered for each row in the table. The Comment field is unused in the configuration.

Do you need a Scan Table? The Internal Channel Scan Table is similar to the Master Channel, but it only refers to polls to the ISaGRAF RTU. In some cases it may not be necessary to include scans to read the ISaGRAF data. To explain why, it is necessary to examine the relationship between the ISaGRAF RTU and the RTDB.

The ISaGRAF RTU (logic, I/O, and data) exists as a separate task inside the RediGate. Similar to an external RTU, the ISaGRAF RTU contains data. In order to get this data into an RTDB, it must be scanned by the Scan Table – even though it resides inside the same hardware. The reason for this is to have a consistent Channel structure for all RTUs. For a more complete explanation, see the section ISaGRAF Channel Functional Elements.

If the ISaGRAF RTU does not contain data that is necessary to store in an RTDB, then it is not necessary to define any polls to the ISaGRAF RTU. As one example, the ISaGRAF logic may be written with special ISaGRAF functions such as DMOV or ISAMOV to move data from ISaGRAF or an RTDB, or from one RTDB into another. Or, the ISaGRAF logic may simply act on data from other RTDB's, such as sending a Publish message or an E-mail. In these cases, the data contained within the ISaGRAF RTU may not need to be scanned by the Internal Channel in order to read it into the ISaGRAF RTDB, and thus no Scan Table rows need to be defined.

Also note that other units under the Internal Channel (Status RTU, Virtual RTU, Segment RTU) do not need to be polled for data. They reside under the Null Circuit, and their data simply appears in their own RTDBs automatically without being scanned.

Null Circuit

A Null Circuit object defines placeholder in the configuration, under which one or more Status/Control, Virtual, Segment, or Internal Master field units are defined for an Internal Channel.

Attributes	Function
Object Type	NullCircuit
Parent(s)	System Clients Master Channels Internal Channel
Instance	Must be 0 or 1. (However, the Virtual Circuit must be instance 0, so if used with a Null Circuit, the Null Circuit must be instance 1.)

The Null Circuit should have at least one child Filed Unit object defined under it.

Properties	Values
Circuit Type	Placeholder for the circuit type as 'Null Circuit'.

Virtual Field Unit

A Virtual Field Unit object allows an additional Real-time Database (RTDB) to be defined for internal storage of data. This may be used as a data repository for the ISaGRAF program logic to store data values, and/or the RTDB may serve as the source of data for a Slave Channel definition.

> Note: The Virtual Field Unit will not automatically be marked alive for purposes of reporting via an MQTT client. To make sure the Virtual Field Unit is marked alive, one of the following needs to be done:
>
> - In the Internal Master Channel, define one scan for the Virtual unit (use Poll Record 1). Make sure the Scan Period is very long (longer than the Scan Effective Limit). Even though the poll will be marked as a timeout, the Virtual Unit will be marked alive because the Scan Table completes its cycle.
> - In the Internal Master Channel, if there is an Internal Master unit being scanned in addition to the Virtual unit, the Virtual unit will also be marked as alive. (This occurs because the Channel is required to complete one full scan cycle successfully.)
> - Or you could use a POD "SET RTU STATUS" or ISaGRAF 'setosval' function to set the Virtual unit to an alive state.

Attributes	Function
Object Type	FieldUnitVirtual
Parent(s)	System Clients Master Channels Internal Channel NullCircuit
Instance	Must be between 0 and 255, and should be unique under this Null Circuit among all Status, Virtual, and Internal Master field units.

Properties	Values
Unit Name	Enter the field unit name as an identifier for the Field Unit.
Unit Address	Enter the Field Unit address. This should be a unique address from any other ISaGRAF, Status, or Internal Master field unit addresses on this Internal Channel. *The default address is 3.*
Protocol	Placeholder for field unit protocol type 'Repository Database Unit'.
Com Retries	This parameter is a null field to retain compatibility with other field unit objects. *The only option is 'N/A'.*
Comm Status Holdreg	Enter the **starting** holding register to contain the communication status for this field unit. Typically, this field is unused for the Virtual Unit. Each Comm Status takes 5 registers, beginning at the register configured in this parameter. The Comm Status Holdreg for each field unit in a configuration must be defined such that the five registers do not overlap other registers being used. If the register is defined in the 30,xxx address range, the status values will be stored in the local device's RTDB (i.e., the RTDB defined as a child object to this ISaGRAF Field Unit. If the register is in the 40,xxx range, the values will be stored in the Status/Control Field Unit RTDB. The Comm Status Holdreg is optional, and can be set to 0 to disable the storage of status registers. See the section Communication Status Registers, for a description of the five Comm Status Register contents.

Produce RBEs	Select this option to determine whether to produce a Report by Exception (RBE) flag when data in this unit's RTDB changes. *In the RTDB, for every data point, there are potentially 4 RBE flags associated with every data point. When the data point changes, the RBE flags are set. These flags are used to determine when new data needs to be reported to the HCP.*
No Polls	This parameter is a null field to retain compatibility with other field unit objects. *The only option is 'N/A'.*

Internal Master Field Unit

The Internal Master is a special type of Field Unit which is designed to provide an easy mechanism for collecting and consolidating data from any other Field Unit RTDBs into the RTDB associated with the Internal Master unit. In theory, it operates like a Master Channel Field Unit, which polls data from an external device; but the Internal Master operates only to take data from one RTDB to another.

Attributes	Function
Object Type	FieldUnitInternalMast
Parent(s)	System Clients Master Channels Internal Channel NullCircuit
Instance	Must be between 0 and 256, and should be unique under this Null Circuit among all Status, Virtual, and Internal Master field units. The Internal Master Field Unit should have a Modbus RTDB child object defined under it (see page).

Properties	Values
Unit Name	Enter the field unit name.
Unit Address	Enter the field unit address. This should be a unique address from any other ISaGRAF, Status, Virtual, or Internal Master field unit addresses on the Internal Channel. *The default address is 5.*
Protocol	Select the Protocol type for the Internal Master field unit. The Protocol selection determines the rules by which a host can access the data in this Virtual Master's RTDB, via attachment to a Slave Channel. Available Protocol types are: • ***Read/Write Internal RTU, not to DBM***. This allows a host to both read data from the RTDB and write data directly to the device configured in the Source Channel/RTU. • ***Read/Only Internal RTU***. This allows a host to read data from the RTDB, but writes are not allowed. Data in the RTDB can be changed by ISaGRAF or by Internal Master polls from other RTDB locations, but not by an external host connected via Slave Channel. • ***R/W RTU or to DBM if no Remapped Poll Record***. This allows a host to read data from the RTDB and write data directly to the device configured in the Source Channel/RTU if there are matching poll records; otherwise, write data is stored directly into this Internal Master's RTDB registers.
Com Retries	Enter the number of communication retries after a failed poll attempt. *If a poll attempt fails, poll will be sent again up to the configured number of "Com Retries" before the field unit is declared failed.*
Comm Status Holdreg	Enter the **starting** holding register to contain the communication status for this field unit. Each Comm Status takes 5 registers, beginning at the register configured in this parameter. The Comm Status Holdreg for each field unit in a configuration must be defined such that the five registers do not overlap other registers being used. If the register is defined in the 30,xxx address range, the status values will be stored in the local device's RTDB (i.e., the RTDB defined as a child to this Field unit). If the register is in the 40,xxx range, the values will be stored in the Status/Control Field Unit RTDB. The Comm Status Holdreg is optional, and can be set to 0 to disable the storage of status registers. See the section Communication Status Registers, for a description of the five Comm Status Register contents.
Produce RBEs	Select this option to determine whether to produce a Report by Exception (RBE) flag when data in this unit's RTDB changes. *In the RTDB, for every data point, there are potentially 4 RBE flags associated with every data point. When the data point changes, the RBE flags are set. These flags are used to determine when new data needs to be reported to the HCP.*

Poll Table	Click the **Edit Table** button to define the Modbus polls to be sent to this unit. Note that the Poll Table only defines how the Modbus protocol is defined to operate for each set of data defined in the polls. The Poll Table doesn't actually do any of the polling itself. If you want any of these polls to be sent to the Field Unit on a regular basis, it should be referenced in one or more Scan Table entries in the Master Channel. **Src Chan** – Enter the Channel number of the RTDB containing the source data. This is either a Master Channel or Internal Channel instance number that must be defined under the Master Channels placeholder. **Src RTU** – Enter the Field Unit number of the RTDB containing the source data. This is the Unit Address configured in the properties of the field unit, not the instance number of the field unit ACE object, if they are different. **Src Data** – Enter the source register number for the starting register in the Field Unit's RTDB to begin retrieving data. **Src Type** - Enter the data type of the data being requested. See below for a discussion of Src Type options in the Master Channel. ***Src Count*** – Enter the number of registers to retrieve. *The maximum number allowed in any poll is the same as for Modbus polls: 2000 Boolean registers, 125 registers of 16-bit type, or 62 registers of 32-bit type. The Count includes the number of register values being requested, starting at the Src Data register.* ***Dest Data*** – Enter the starting destination register within this Internal Master's RTDB to place the polled data. Ensure that there is a large enough quantity of registers in the RTDB to store the count. *The destination register type should be chosen based on the Source Format of the data. Booleans should be stored into Boolean RTDB registers. 16-bit values should be stored into 16-bit RTDB registers. 32-bit values or 16-bit pair values should be stored into 32-bit RTDB registers.* When reading data from internal databases, the Internal Master also reads the quality flag status for each point, and individually sets the quality flags on the destination data to match the source point status.

Discussion on Source Type

Several **Src Type** options are provided in the Internal Master Field Unit. These provide a number of unique capabilities for copying and transforming data from one RTDB location to another.
The following table gives a list of the Internal Master **Src Type** options, and an explanation of how they are used.

Src Type	Source data	Dest data	Meaning
Boolean	Boolean	Boolean	Single on/off bit occupies a register
8 bit			Take lower 16-bits of an integer register.
16 bit	16 bit	16 bit	Typical 16-bit register type.
24 bit	32-bit	32-bit	Take lower 3 bytes of a 32-bit integer register
32 bit	32 bit	32 bit	
Short String	STRING32	STRING32	
Long String	STRING256	STRING256	
UTF String			
Event			
Double-Float	64 bit	64 bit	
Copy 16-Bit pairs into 32-Bit Regs	16 bit	32 bit	Registers are taken as pairs, and the Src Count should be the numbers of pairs of registers (32-bit entities). (Source registers are little-endian.)
Swap 16-Bit pairs into 32-Bit Regs	16 bit	32 bit	Registers are taken pairs, and the Src Count should be the numbers of pairs of registers (32-bit entities). (Source registers are big-endian.)
Split-Copy 32-bit Regs into 16-bit-Pair Regs	32 bit	16 bit	32-bit register is broken into pairs of 16-bit registers
Split-Swap 32-bit Regs into 16-bit-Pair Regs	32 bit	16 bit	same as above, but swapped
Boolean-Src to 16-Bit-Dest Hi to Lo Bits	Boolean	16 bit	Boolean source registers are taken in groups of 16, with the first register becoming the most-significant bit (MSB) in the 16-bit value.
16-Bit-Src to Boolean-Dest Hi to Lo Bits	16 bit	Boolean	16 bits in source register(s) are placed sequentially to Boolean registers, in the order of most-significant bit (MSB=1st destination register) to least-significant bit in each 16-bit word.

16-Bit-Src to Boolean-Dest Lo to Hi Bits	16 bit	Boolean	16 bits in source register(s) are placed sequentially to Boolean registers, in the order of least-significant bit (LSB=1st destination register) to most-significant bit in each 16-bit word.

The following source types reverse the direction of data from the other types listed above, allowing data to be copied from the Internal Master RTDB to a different Field Unit's RTDB location. Be sure to keep in mind that for these data types, the "Src" (source) and "Dest" (destination) data locations are reversed. Thus:

Src Chan/RTU/Data are the destination locations to store data in the other Field Unit.

Dest Data is the source RTDB location in the Internal Master field unit.

The source and destination data types should be the same.

Src Type	Meaning
Trigger Move Data from Local to Src Data	Choose this option to copy data from the Internal Master RTDB (Dest Data location) into another RTDB (Src Data location), based on a trigger value. The Trigger register is the last register in the specified set. When the Trigger register is set to a non-zero value by any means (external host, ISaGRAF, POD, etc.), then the full Count of registers are copied from the Dest registers to the Source Chan/RTU/Data register location, and the Trigger register is set back to zero automatically. Thus, if using a **Dest Data** register of 40,001 (in the Internal Master), and a **Src Count** of 17 registers, then the Internal Master RTDB register 40,017 is used as a trigger for the move operation.
Trigger Write Data from Local Data to Src Chan/RTU	Choose this option to write data from the Internal Master RTDB to a device on a Source Channel/RTU address, based on a trigger value. The Trigger register is the last register in the specified set. When the Trigger register is set to a non-zero value by any means (external host, ISaGRAF, POD, etc.), then the all of the registers are written from the Dest registers to the Source Chan/RTU/Data register location using the device's protocol, and the Trigger register in the local RTDB is set back to zero automatically.
Always Move Data from Local to Src Data	Choose this option to constantly copy data from the Internal Master RTDB (Dest Data location) into another RTDB (Src Data location). No trigger register is used.
Always Write Data from Local Data to Src Chan/RTU	Choose this option to constantly write data from the Internal Master RTDB (Dest Data location) to a device on a Source Channel/RTU address/Src Data register whenever this poll is scanned by the Internal Master Channel. No trigger register is used.

The final Internal Master "Source Type" is a special function that tells the Internal Master process to run a POD logic routine. This POD program is run in the sequence of the scans that are triggered by the Internal Master. The POD program must complete before the Scan Table can move on to the next scan.

Run POD	[Src Count] Src Count Column value is Pod_Index. Each POD has a unique instance number, and when this Poll Table entry is trigged, the numbered POD routine runs in its entirety before control is returned to the Internal Channel for running subsequent polls.

The POD object in the ACE configuration holds a set of programming instructions which can be used to manipulate or make decisions on data stored in RTDBs. Up to 9999 POD modules can be configured per Internal Master Channel, which are called by a Poll Table of an Internal Master RTU (which in turn is triggered by a Scan Table entry in the Internal Master Channel).

Status Field Unit

A Status Field Unit object may be used to contain unique information for internal communication diagnostics. Field units defined under Master Channels and the ISaGRAF field unit contain a parameter for storing communication status (Comm Status Register). If the Comm Status Register for any Field Unit is configured with a starting address in the 40,xxx address range, its status values are stored in the RTDB defined for this Status Field Unit. This allows all communication status information to be stored in one place, if desired in the system design.

However, if the Comm Status Register for any Field Unit is defined with a starting address in the 30,xxx address range, the communication status values for that device are stored in the RTDB for that Field Unit rather than the common Status Field Unit RTDB. In that case, the Status Field Unit is not necessary and may be omitted from the configuration.

See the section Communication Status Registers for a description of the communication status registers stored for Field Units.

Attributes	Function
Object Type	FieldUnitSysCtrl
Parent(s)	System Clients Master Channels Internal Channel NullCircuit
Instance	Must be between 0 and 255, and should be unique under this Null Circuit among all Status, Virtual, and Internal Master field units. Only one Status Field Unit should be included in a configuration.

The Status Field Unit should have a Modbus RTDB child object defined under it (see page).

Properties	Values
Unit Name	Enter the field unit name as an identifier.
Unit Address	Enter the field unit address. This should be a unique address from any other ISaGRAF, Virtual, or Internal Master field unit addresses on the Internal Channel.
Protocol	Placeholder for field unit protocol type 'System Control & Status Unit'.
Com Retries	This parameter is a null field to retain compatibility with other field unit objects. *The only option is 'N/A'.*
Comm Status HoldReg	This parameter is a null field to retain compatibility with other field unit objects. It should be set to 0.
Produce RBEs	Select this option to determine whether to produce a Report by Exception (RBE) flag when data in this unit's RTDB changes. *In the RTDB, for every data point, there are potentially 4 RBE flags associated with every data point. When the data point changes, the RBE flags are set. These flags are used to determine when new data needs to be reported to the HCP.*
No Polls	This parameter is a null field to retain compatibility with other field unit objects. *The only option is 'N/A'.*

Communication Status Registers

The section Status Field Unit describes the Status Control RTU, which stores communication statuses for field units. The five status values contain the following data for each unit:

1st Register

> ***Bit 0,1,2*** Communication status to field unit. A register value of 0 indicates failed communication, and 7 indicates good communication..
>
> ***Bit 7*** Unit is disabled.

2nd Register Percent (%) Communication throughput to field unit. Throughput = (Total Polls - # Timeouts - # Bad Data polls) / Total Polls. Range is 0 - 1000 scaled, so that a value of 987 = 98.7 %.

3rd Register Total Polls. Total number of polls sent since last restart. When this register reaches 65,000 it rolls over to zero, and the 2nd, 4th, and 5th status bytes are also reset to zero.

4th Register # Timeouts. Number of polls receiving no response since last reset.

5th Register # Bad Data polls. Number of CRC or data errors to polls since last reset.

The RTDB for the Status/Control field unit must include an adequate number of holding registers to contain all the Comm Status registers for all defined Field Units, and these 5 Comm Status registers for each device must not overlap each other.

Segment Field Unit

A Segment Field Unit object allows Segment databases (RTDB) to be defined with a more flexible structure than most RTDBs. Like the Virtual unit, the Segment Field Unit is simply a data repository and does not require any Internal Channel scans to be defined. The Segment unit is used to create a more granular list of registers with a mix of different data types.

Attributes	Function
Object Type	FieldUnitSegment
Parent(s)	System Clients Master Channels Internal Channel NullCircuit
Instance	Must be between 0 and 255, and should be unique under this Null Circuit among all Status, Virtual, and Internal Master field units. The Segment Field Unit should have a Segment RTDB child object defined under it.

Properties	Values
Unit Name	Enter the field unit name as an identifier for the Field Unit.
Unit Address	Enter the Field Unit address. This should be a unique address from any other ISaGRAF, Status, Virtual, or Internal Master field unit addresses on this Internal Channel.
Protocol	Placeholder for field unit protocol type 'Segment Database Unit'.
No Comms	Unused
Not Used	Unused
No Polls	Unused

Segment RTDB

A Segment RTDB (Real Time DataBase) defines the size of the virtual database reserved for the Segment Field Unit. The Segment RTDB contains some significant differences from other RTDB objects. It is more flexible in size and construction, and multiple Segment RTDBs may be configured under a single Segment Field Unit. The Segment RTDB also allows deadbands to be defined in the database by point number, rather than through a separate field-based Deadband object.

There are several ISaGRAF functions that allow data to be pushed and retrieved from a Segment point index, to publish a particular Segment RTDB, etc.

The collection of Segment RTDB objects together create a single RTDB as might exist under other Field Unit objects. The Segment RTDB is indexed with only a Point Count, not requiring a register address to be configured. The table below shows the comparison between a set of Segment RTDB objects as compared with a similarly configured RTDB for other field units.

Src Type	Source data	Dest data	Meaning
Boolean	Boolean	Boolean	Single on/off bit occupies a register
8 bit			Take lower 16-bits of an integer register.
16 bit	16 bit	16 bit	Typical 16-bit register type.
24 bit	32-bit	32-bit	Take lower 3 bytes of a 32-bit integer register
32 bit	32 bit	32 bit	
Short String	STRING32	STRING32	

Long String	STRING256	STRING256	
UTF String			
Event			
Double-Float	64 bit	64 bit	
Copy 16-Bit pairs into 32-Bit Regs	16 bit	32 bit	Registers are taken as pairs, and the Src Count should be the numbers of pairs of registers (32-bit entities). (Source registers are little-endian.)
Swap 16-Bit pairs into 32-Bit Regs	16 bit	32 bit	Registers are taken pairs, and the Src Count should be the numbers of pairs of registers (32-bit entities). (Source registers are big-endian.)
Split-Copy 32-bit Regs into 16-bit-Pair Regs	32 bit	16 bit	32-bit register is broken into pairs of 16-bit registers
Split-Swap 32-bit Regs into 16-bit-Pair Regs	32 bit	16 bit	same as above, but swapped
Boolean-Src to 16-Bit-Dest	Boolean	16 bit	Boolean source registers are taken in groups of 16, with the first register becoming the most-significant bit (MSB) in the 16-bit value.
16-Bit-Src to Boolean-Dest	16 bit	Boolean	16 bits in source register(s) are placed sequentially to Boolean registers, in the order of most-significant bit (MSB=1st destination register) to least-significant bit in each 16-bit word.

The instance number and point count for each Segment RTDB implicitly defines the data addresses that can be used when referring to data values in the Segment RTU. Note that the registers in Segment RTDBs are defined in blocks of (multiples of) 300 points per instance number. If one Segment# includes more than 300 points, as with Segment #2 in the above example, you must not skip those Segment RTDB instance numbers to avoid an address conflict.

Attributes	Function
Object Type	RtdbSegmentFdb
Parent(s)	System Clients Master Channels Internal Channel NullCircuit FieldUnitSegment
Instance	Must be between 1 and 255.

Properties	Values
Produce RBEs	Select this option to determine whether to produce a Report by Exception (RBE) flag when data in this unit's RTDB changes. In the RTDB, for every data point, there are potentially 4 RBE flags associated with every data point. When the data point changes, the RBE flags are set. These flags determine when new data needs to be reported to an MQTT server, HCP, or other hosts supporting RBE.
Segment Definition	Click the **Edit Table** button to edit the details of the RTDB definition. **Point Count** – Enter the number of data points of this type to be allocated space in the database. **Point Format** – Select the point data format: • ***UINT8*** – Unsigned 8-bit integer (0 to 255) • ***SINT16*** – Signed 16-bit integer (-32,768 to 32,767) • ***UINT16*** – Unsigned 16-bit integer (0 to 65,535) • ***SINT32*** – Signed 32-bit long integer • ***UINT32*** – Unsigned 32-bit long integer • ***REAL32*** – IEEE floating point (32-bit) **Deadband** – Enter the deadband value for the points defined on this row, which is the amount a value in the RTDB can change before it will be flagged as an RBE. *Deadband value is entered as an integer or floating point value, which is handled as a 15-character (max) string. If IEEE floating point format is used, its entry must include a decimal point (such as 11.0).* **Comment** - Optional column, allowing a descriptive comment to be entered for each row in the table. The Comment field is unused in the configuration.

DirectorPOD

Elecsys provides a separate manual to describe how PODs are configured and to explain each of the many function codes available. See the Red iGate POD Programming Manual for further instructions.

Virtual Circuit

A Virtual Circuit object defines an internal communications path to one ISaGRAF field unit from an Internal Channel. The Virtual Circuit is simply a placeholder, designed to match the operation of a normal Async Circuit.

Attributes	Function
Object Type	VirtualCircuit
Parent(s)	System Clients Master Channels Internal Channel
Instance	Must be 0

The Virtual Circuit should have one ISaGRAF Field Unit child object defined under it.

Properties	Values
Circuit Type	Placeholder for circuit type of 'Virtual Circuit'.

ISaGRAF Field Unit

An ISaGRAF Field Unit object is used to run a protocol task that allows an ISaGRAF logic program to run in the RediGate, and defines parameters for how data is read from and written to the ISaGRAF unit.

The RediGate allows for running two independent ISaGRAF logic programs simultaneously. In this case, the Internal Channels should be defined as instance numbers 14 and 15, with each channel including a Virtual Circuit and ISaGRAF Field Unit objects. In either case, to allow the ISaGRAF Workbench program to download and monitor the ISaGRAF logic program(s), the ISaGRAF Field Unit definition must be attached to a Slave Channel definition. See the section Slave Channels for setting up the Slave Channel.

Attributes	Function
Object Type	FieldUnitIsagraf
Parent(s)	System Clients Master Channels Internal Channel VirtualCircuit
Instance	Must be 0

The ISaGRAF Field Unit should have a Modbus RTDB child object defined under it (see page).

Properties	Values
Unit Name	Enter the field unit name. *Unit name is displayed in diagnostic menus and in an HCP diagnostic screen.*

Unit Address	Enter the actual field unit address which will be used for the ISaGRAF Field Unit. This Modbus unit address will be referenced under a Slave Channel, from which the ISaGRAF workbench will communicate to the logic program. This should be a unique address from any other Status, Virtual, or Internal Master field unit addresses on the Internal Channel. *Valid addresses 1 to 255. Typically address 1 is used for one ISaGRAF unit (often configured in Channel 15), but if using two ISaGRAF RTUs in the same configuration one of them must be configured for address 123.*
Protocol	Placeholder for the protocol type as 'IsaGraf Logic Unit'.
Com Retries	Enter the number of communication retries after a failed poll attempt. *If a poll attempt fails, it will try again up to the configured number of "Com Retries" before the field unit is declared failed.*
Comm Status Holdreg	Enter the **starting** holding register to contain the communication status for this field unit. Each Comm Status takes 5 registers, beginning at the register configured in this parameter. The Comm Status Holdreg for each field unit in a configuration must be defined such that the five registers do not overlap other registers being used. If the register is defined in the 30,xxx address range, the status values will be stored in the local device's RTDB (i.e., the RTDB defined as a child object to this ISaGRAF Field Unit. If the register is in the 40,xxx range, the values will be stored in the Status/Control Field Unit RTDB. The Comm Status Holdreg is optional, and can be set to 0 to disable the storage of status registers. See the section Communication Status Registers, for a description of the five Comm Status Register contents.
Produce RBEs	Select this option to determine whether to produce a Report by Exception (RBE) flag when data in this unit's RTDB changes. *In the RTDB, for every data point, there are potentially 4 RBE flags associated with every data point. When the data point changes, the RBE flags are set. These flags are used to determine when new data needs to be reported to the HCP.*
Poll Table	Click the **Edit Table** button to define the Modbus polls to be sent to this ISaGRAF field unit. Note that the Poll Table only defines how the Modbus protocol is defined to operate for each set of data defined in the polls. The Poll Table doesn't actually do any of the polling itself. If you want any of these polls to be sent to the ISaGRAF Field Unit on a regular basis, it should be referenced in one or more Scan Table entries in the Internal Channel. **Source Register** – Enter the source register in the ISaGRAF RTU to begin polling data. This will be a Modbus register defined on an I/O board definition in the ISaGRAF logic program. **Source Format** – Enter the data type of the data being requested. See below for a discussion of Source Format options. *See the section Modbus/Open Modbus TCP Field Unit for a full discussion on Source Formats. For the ISaGRAF RTU, typically the following Source Formats should be used:* • *Boolean – for boards defining Boolean data* • *16 bit register (HL) – for boards defining 16-bit data* • *32 Bit B/B Endian (HLhl) – for ISaGRAF boards defining 32-bit data* **Count** – Enter the number of registers to poll. *The maximum number allowed in any poll is 2000 Boolean registers, 125 registers of 16-bit type, or 62 registers of 32-bit type.* *The Count includes the number of entities being requested, based on the Source Format type listed above.* *For instance, if polling 10 sets of 32-bit data which occupy pairs of 16-bit registers (one of the "16 Bit Pair" types of data), the Count should be 10 for the number of entities (not 20 for the number of registers being requested).* **Destination Register** – Enter the starting destination register within the RTDB (Real Time Data Base) to place the polled data. *The Destination Register type should be chosen based on the Source Format of the data.* *Booleans should be stored into Boolean RTDB registers.* *16-bit values should be stored into 16-bit RTDB registers.* *32-bit values or 16-bit pair values should be stored into 32-bit RTDB registers* **Comment** - Optional column, allowing a descriptive comment to be entered for each row in the table. The Comment field is unused in the configuration.

Load/Store ISaGRAF Defaults

The Load/Store object defines a separate download file, containing parameter values used by the ISaGRAF STORE16 and STORE32 boards. The STORE boards are pseudo boards containing variables that are stored in the Flash file system (see the *Elecsys ISaGRAF Manual* for more details on these boards).

When an entry in the STORE16 or STORE32 board is written (or a variable attached to one of the STORE boards), ISaGRAF automatically creates a file in the file system with the name **ls***aabbcccc* (first character is the letter "L"), where *aabbcccc* is identical to the ISaGRAF file **is***aabbcccc*. Defining the Load/Store object in the ACE configuration defines default values to be stored in the STORE board if it doesn't already exist. This allows the ISaGRAF application to be written generically, and yet to act differently for individual units based on the device-specific values contained in STORE boards, as configured in the ACE Load/Store object.

After the **ls***aabbcccc* file is created or downloaded to the RediGate, its values may be changed by the ISaGRAF program operation. Downloading the file from ACE again would overwrite any values that have subsequently been stored in the LoadStore file.

Attributes	Function
Object Type	LoadStore
Parent(s)	System Clients Master Channels Internal Channel VirtualCircuit FieldUnitIsagraf
Instance	Must be 0
UFF External	Checkbox should be enabled.

Properties	Values
Parameter Table	Click the **Edit Table** button to edit the default values contained in the Load/Store parameter table. **Parameter 1** through **Parameter 16** – Enter the value for each parameter defined in the ISaGRAF STORE16 or STORE32 board. Multiple rows defined in the Parameter Table correspond to multiple sequential instances of the STORE boards (in the order of instances defined in the ISaGRAF application). **Comment** - Optional column, allowing a descriptive comment to be entered for each row in the table. The Comment field is unused in the configuration. When uploading the ACE configuration, the **ls***aabbcccc* file will be created and uploaded separately at the same time.

TextStore Object

A TextStore object defines a separate download file in the configuration, containing default text strings which may be read and used by the ISaGRAF program logic. The TextStore file is created with the name **st***aabbcccc*, where *aabbcccc* is identical to the ISaGRAF file **is***aabbcccc* to which it is attached.

This allows an ISaGRAF program to be written that can take user-configured text entries that are specific to each RediGate. This allows the ISaGRAF application to be written generically, and yet to act differently for individual units based on the device-specific text values contained in TextStore boards, as configured in the ACE Load/Store object.

After the **st***aabbcccc* file is created or downloaded, its values may be changed by the ISaGRAF program operation. Downloading the file from ACE again would overwrite any values that have subsequently been stored in the TextStore file.

Attributes	Function
Object Type	TextStore
Parent(s)	System Clients Master Channels Internal Channel VirtualCircuit FieldUnitIsagraf
Instance	Must be 0
UFF External	Checkbox should be enabled.

Properties	Values

Text Table	Click the **Edit Table** button to edit the default values of the TextStore object. **Text** – Enter the text for each field. The text fields are limited to 16 characters per string. **Comment** - Optional column, allowing a descriptive comment to be entered for each row in the table. The Comment field is unused in the configuration. When uploading the ACE configuration, the **st***aabbcccc* file will be created and uploaded separately at the same time.