

RTS PROTOCOL DESCRIPTION

Introduction

The RTS protocol allows any program to execute SCL controls on an U.P.M.A.C.S. station via a socket TCP connection. Special SCL Commands and functions are used to read parameters supplied with the request, and to send responses to the RTS client.

▪ Connection Mechanism

RTS supports two distinct connection mechanisms:

Single Control Connections

A separate TCP connection is used for each control. The connection is established by the client, the request to execute the control is sent, and any replies are received. When the control has terminated, the server drops the connection.

Multiple Control Connections

A single TCP connection can be used to execute multiple controls. The connection is first established by the client. Requests to execute controls can then be sent at any time, and any replies are received. It is not necessary to wait for one control to finish before executing the next; multiple controls can be running concurrently. When a control terminates, the client is notified via a special "control done" packet. The connection will remain open until the client drops it, the server shuts down (U.P.M.A.C.S. quits), or a badly formatted request packet is received.

The RTS client can use whichever mechanism is more convenient. Multiple connections of either type can be open simultaneously.

▪ Request Parameters

An RTS request can have any number of parameters, which will be available to the SCL control. The parameters can contain arbitrary data, and are accessed as strings from within SCL. (SCL strings can contain any characters, including NULL characters.) The decoding of the parameter data is left up to the SCL control.

SCL supports encoding and decoding of numerical data using a variety of data formats, including signed and unsigned single-byte and multi-byte values in both low-high and high-low byte ordering. It also supports BCD (binary coded decimal) encoding, as well as numbers written out in ASCII characters using decimal, hexadecimal, binary, or octal base.

To transmit integer or fixed point numbers, the RTS client can choose whichever encoding is most convenient. For floating point numbers, however, SCL only supports decimal numbers written out in ASCII characters. Exponential notation ("1.2E+03", etc.) is *not* supported.

- **Reference Number**

The RTS client must specify a reference number for each RTS request. The reference number is not used by U.P.M.A.C.S., but it is included in any reply or error message that pertains to that request. The client may use the reference number as it wishes. The reference number does not have to be unique, but please note that if you execute more than one control simultaneously using a multiple control connection, the reference number is the only way to determine which control sent a specific reply.

- **RTS Replies**

An RTS control can send one or more replies or error messages to the client. Each reply or error message contains a single block of data, formatted by the SCL control in any fashion it pleases.

The RTS server may also send error messages. These error messages will contain an error code and a description of the error. For multiple control connections, a special "control done" message is sent when a control terminates. "Control done" messages have no data associated with it.

RTS Protocol Description

U.P.M.A.C.S. accepts RTS (and other) socket connections on TCP port 8700. RTS connections are distinguished from other types of connections using the packet opcode field. The packet opcode field is also used to distinguish between single control and multiple control connections.

- **Length Encoding**

U.P.M.A.C.S. has a special way of encoding data lengths. Since all characters, including the NULL character, are valid for RTS parameters and replies, data lengths are used rather than NULL-terminated strings. Data length is limited to 65535 characters, and is encoded into a 32-bit value as follows:

$$\text{encoded_length} = \text{length} \mid (\text{length} \wedge 0x5555) \ll 16$$

- **Request Packet Format**

An RTS request packet has the following format:

| Field Length | Field Content |
|--|---|
| 8 bits | Opcode: 0x04 for single control connections 0x05 for multiple control connections |
| 32 bits | Reference number |
| 32 bits | Encoded length of program tag |
| Tag length × 8 bits | Tag of the program to be executed |
| 32 bits | Number of optional parameter blocks |
| Optional parameter blocks: (one for each parameter) | |
| 32 bits | Encoded length of parameter data |
| Data length × 8 bits | Parameter data |

The opcode of the first packet sent to the server after opening a connection determines whether the connection is a single or a multiple control connection.

- **Reply Packet Format**

An RTS reply packet has the following format:

| Field Length | Field Content |
|----------------------|------------------------------|
| 8 bits | Opcode: 0x03 |
| 32 bits | Reference number |
| 8 bits | Error code (see below) |
| 32 bits | Encoded length of reply data |
| Data length × 8 bits | Reply data |
| 8 bits | Terminating NULL: 0x00 |

The terminating NULL can be used by the client to read the data without decoding the data length. Note, however, that an SCL control may send data with embedded NULL characters in it, since all characters are legal reply data.

For error codes 0x00 (no error) and 0x05 (user error), the format and meaning of the reply data is determined by the SCL control. For all other error codes, the reply data is simply a human-readable error message from the server.

Error Codes

The following error codes may be returned in a reply packet:

- 0x00 *No error*: The reply was sent by the SCL control using the `RTSEND` command.
- 0x01 *Bad request packet format*: The request packet was badly formatted. If this error code is sent by the server, the connection is dropped immediately, even if it is a multiple control connection.
- 0x02 *Program not found*: There is no program with the requested tag in the current station.
- 0x03 *Request denied*: The server is not accepting RTS requests. The U.P.M.A.C.S. Operate System will only accept RTS requests if you enable insecure remote control in the Network Security Settings.
- 0x04 *Program error*: An SCL program error occurred while executing the control.
- 0x05 *User error*: The reply was sent by the SCL control using the `RTSErrorR` command.

- **“Control Done” Packet Format**

A multiple control connection “control done” packet has the following format:

| Field Length | Field Content |
|--------------|------------------|
| 8 bits | Opcode: 0x02 |
| 32 bits | Reference number |

Writing SCL Programs for RTS Controls

The SCL language incorporates one function and two commands for use with RTS.

Use the `RTSPRM$` function to access the parameters specified in the request packet. Use the `RTSEND` command to send a reply packet with error code 0x00. Use the `RTSErrorR` command to send a reply packet with error code 0x05.

You can use the string conversion functions if you need to decode numerical data from the RTS parameters, or if you need to encode numerical data for a reply. You cannot use user interface (message or dialog) command in programs for RTS controls.

For more information on writing programs for RTS controls, please see *RTS Controls* in the SCL Language Reference.