Seagull - Developer

Table of contents

1. Developer documentation

The purpose of this documentation is to explain Seagull interns. It is a summary to help developers make their way through Seagull and not intended to be a definitive developer documentation.

2. Main modules

- **generator-core**: core of Seagull. Contains the main functions.
- **generator-model**: internal model of Seagull to manage protocols, channels and transports.
- **generator-scenario**: model for the scenario
- **generator-traffic**: traffic model. Management of the call context, of the read/write on transport channel, of the execution of the calls.
- **protocol-data**: definition of generic data used by the generator
- **protocol-frame**: virtual definition of a protocol and its messages. It contains the only objects that will be used by the generator
- **transport-frame**: generic definition of a transport

3. Other modules

- **common**: contains all the .h files to adapt some system includes to the target compilation platform
- **data-log**: deals with response time logs (csv file)
- **exe-env**: execution environment for each protocol
- **external-data**: manage external data files, to insert data into scenario messages during the call
- **generator-common**: utils for the tool (buffer management, id, ...)
- **library-trans-extsctp**: SCTP transport library; daughter class of C_Transport
- **library-trans-ip**: IP v4 & v6 transport transport; daughter class of C_Transport
- **protocol-binary**: binary protocol definition (e.g. Diameter); daughter class of C_ProtocolBinaryFrame, C_MessageFrame
- **statistics**: stats management
- **xml-parser**: parsing the xml files (dictionary, scenario and configuration file)

4. Main objects

- **C_Generator** (module generator-core): Main object
- **C_ReadControl** (module generator-traffic)
- **C_CallControl** (module generator-traffic): Objects for the execution of the scenario
- Protocol objects:
- **C_ProtocolControl** (module generator-model): The only object that knows about the non-virtual message and protocol objects.

5. Threads

There are 5 threads in Seagull:

- One thread that handles the whole traffic,
- One thread for the keyboard,
- One thread for the tool display,
- One thread for the logs,
- One thread for the statistics.

They are handled in generator-core module

6. Protocol and messages

- To implement a text protocol, you need to create a new implementation of C_ProtocolFrame and C_MessageFrame like C_ProtocolBinary and C_MessageBinary
- In those classes, you implement the management of the XML tags and the protocol definition
- You need to add them in C_ProtocolControl to be instantiated by the tool

7. XML parsing

- Handled in module xml-parser
- XML files are parsed by C_XmlParser
- XML data are represented as a tree of elements (based on stl types): (<TAG (FIELD="VALUE")*> <\TAG>)*
- Useful functions in C_XmlData allows to get the name of a tag, retrieve the value of a field or the list (stl) of element under one element.

8. Channel creation

- C_TransIP (module library-trans-ip) implements the transport, based on virtual object C_Transport (module transport-frame)
- Done/Declared in conf file.
- Channel = link between one protocol and one transport, with an open command for the transport

9. Multi-channel policy

Automatic session based: possible to open several channels (using same or different protocols), but only the first channel used can be server.