SGIO Quick Reference

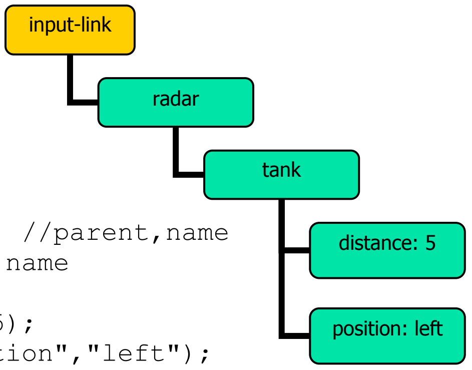
This quick reference shows examples of using the most common SGIO objects and methods. The example io-link structures are derived from TankSoar. This is not meant to be a functional program.

Initialization

```
//For API Soar (i.e. integrated kernel)
sgio::Soar* soar = new sgio::APISoar();
//For SIO Soar (i.e. TSI debug windows)
sgio::Soar* soar = new sgio::SIOSoar("127.0.0.1",6969,true); //IP,port,lockstep

sgio::Agent* agent = soar->CreateAgent("my-agent"); //agent name
agent->LoadProductions("my-agent.soar"); //file name

sgio::WorkingMemory* mem = new sgio::WorkingMemory(agent);
```



Adding WME's to Input-Link

```
//Args for ID's: parent, ID name
sgio::SoarId* radarId = mem->CreateIdWME(mem->GetILink(),"radar"); //parent,name
sgio::SoarId* tankId = mem->CreateIdWME(radarId,"tank"); //parent,name
//Args for Elements: parent, attribute name, attribute value
sgio::IntElement* distance = mem->CreateIntWME(tankId,"distance",5);
sgio::StringElement* position = mem->CreateStringWME(tankId,"position","left");
```

Modifying Existing WME's on Input-Link

```
mem->Update(distance,4); //element, new attribute value
```

Removing WME's on Input-Link

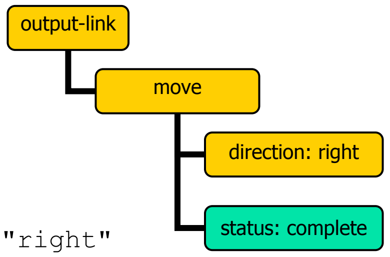
```
mem->DestroyWME(tank); //element to remove; children automatically removed
```

Sending Changes to Soar

```
mem->Commit();
```

Running Agents

```
//To run all the agents on a particular connection for a max of 15 decision cycles
soar->RunTilOutput();
//To run a single agent for a max of 15 decision cycles
agent->RunTilOutput();
```



Checking Output-Link for Commands

```
bool waiting = agent->Commands();
```

Reading Commands from the Output-Link

```
std::auto_ptr<sgio::Command>cmd = agent->GetCommand();
std::string name = cmd->GetCommandName(); //name = "move"
std::string value = cmd->GetParameterValue("direction"); //value = "right"
```

Marking Command as Processed

```
//If everything goes well
cmd->AddStatusComplete();
//If there is an error (i.e. missing attributes)
cmd->AddStatusError();
cmd->AddErrorCode(5); //integer
```

Shutting Down

```
//All other SGIO objects (i.e. agents, commands, elements) are cleaned up automatically
delete mem;
delete soar;
```

Binaries

- simside.lib: Application-side library
- soarside.exe: Remote proxy for Soar (for SIO Soar)
- sgio_shared.lib: Some common code, i.e. messages for SIO Soar

Building an Application

- Application must be compiled multithreaded
- Need to include shared and simside directories for headers
- Need to link simside.lib and sgio_shared.lib
- Need to link soarkernel.lib (version 8.4.5) if using API Soar

Headers to Include (all from simside directory)

- sgio_siosoar.h: SIOSoar class
- sgio_apisoar.h: APISoar class
- sgio_wmemem.h: WorkingMemory, Element classes
- sgio_command.h: Command class
- sgio_agent.h: Agent class

Running an Application

- SIO Soar
  - soarside.exe must be running on target machine so application has something to connect to
  - tcl83.dll and tk83.dll must be accessible by soarside.exe
  - Tcl-8.4.x does not work (due to bug in Tcl)
- API Soar
  - Everything is integrated, should just run

Running soarside.exe

- Command line parameters
  - Port number: default 6969
  - Init file: default soarside-init.tcl
- Init file contents
  - Switch to directory containing Soar
  - source start-soar.tcl
  - Switch to directory containing Agents
- See “Soarside User Documentation” in SGIO docs directory