The TaskBrowser Component

Table of Contents

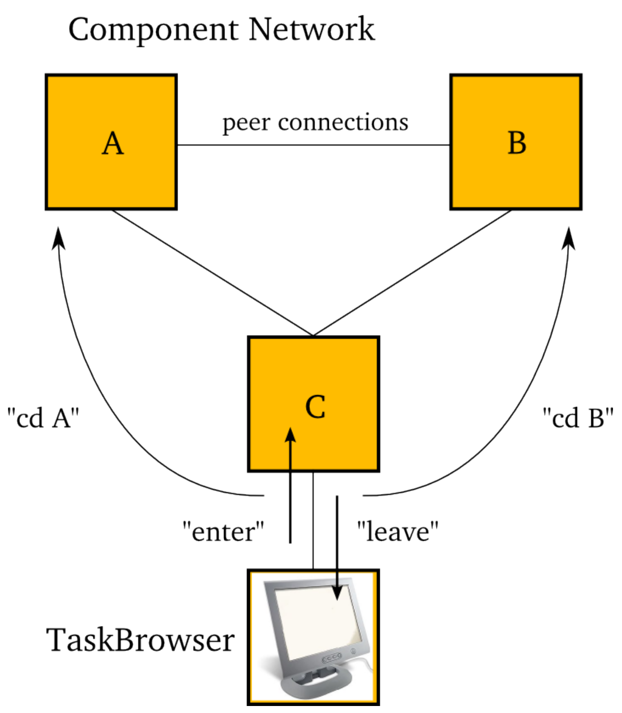
1. Introduction

This document describes the TaskBrowser Orocos component for user interaction with other components. It can visit any component in a given network, query it, use its services, run programs etc.

1.1. Principle

Each Orocos component has a standard interface of properties, ports and operations. It is possible to query and use this interface generically. The TaskBrowser is connected to one component only at a time. It can however be used to 'browse' the network of peer components and visit any component in an application.

The TaskBrowser is a component itself, and it offers a 'window' to another component. When it is connected to another component, it dynamically creates data ports and connects these to the other component. In this way, the TaskBrowser can send data to any component. When the component is left, the ports are disconnected and removed.



The TaskBrowser is initially connected to a component. It can 'enter' a component, which shows a perspective from 'within' the component. it can 'leave' a component which shows a perspective from outside the component. The 'cd' command allows to visit other components.

Figure 1. Task Browsing

There are two possible views on a component: from inside the component, as a program inside the component sees the component interface, or from outside the component, as a peer component sees the component interface. The Taskbrowser can offer the user both views, allowing maximum interactivity. In both views, the component network can be browsed.

2. TaskBrowser Setup

Consult the Component Builder's Manual [http://www.orocos.org/toolchain] for instructions on how to setup and use the TaskBrowser. Typically, you will start the 'deployer' application.

3. TaskBrowser Commands

In addition to giving commands to the components, the TaskBrowser itself can also accept a number of commands. Hit TAB twice to get a list. Most commands accept TAB-completed arguments as well.

Table 1. TaskBrowser Commands

Command	[optional] Argument	Description & Notes
help	-	Display and overview of all the available TaskBrowser commands.
ls	[peer]	Lists the interface and status of a Component. Without peer, the current visited component is shown.
cd	peer	Change to a peer of the current component.
cd ..	-	Change to the previously visited peer.
help <servicename>	-	Display the interface of a service of the current component. Use *this* to display the interface of the current component itself. To see only one operation, use *help <servicename.operationname>* or just *help <operationname>*.
.types	-	Lists all types known to this process.
.services	-	Lists all services known to this process.
.plugins	-	Lists all plugins known to this process.
enter	-	Interprete commands in the context of the current component, as if the current component was issuing them.
leave	-	Interprete commands in the context of the TaskBrowser, as if an external component is communicating with the current component.
list	[script-name] [row]	List the source of a loaded program or state machine script. If the script-name is omitted, list the last list-

Command	**[optional] Argument**	**Description & Notes**
		ed script again. An optional row argument can be given, otherwise, the current point of execution is shown.
trace	[script-name]	Follow the point of execution of a loaded program or state machine. The taskbrowser will display the script source and point of execution when it changed and the user pressed [Enter] at the console prompt. When no arguments are given, all scripts are traced.
untrace	[script-name]	No longer trace a script.
.light	-	Inform the TaskBrowser that your console window has a light background.
.dark	-	Inform the TaskBrowser that your console window has a dark background.
.nocolors	-	Inform the TaskBrowser to disable all coloring.
.record	macro-name	Start a new macro which will be saved as 'macro-name.ops' which will contain an exported function with that name. You will receive a new prompt and are required to type in scripting syntax ('do', 'set', 'if',...)
.cancel	-	Cancel the recording of the current macro. This brings you back to the standard TaskBrowser prompt.
.end	-	Finish and store the current macro. See .loadProgram to load this macro into the current TaskContext. It will appear as a command which takes no arguments.

Command	**[optional] Argument**	**Description & Notes**
quit	-	Exit the TaskBrowser.