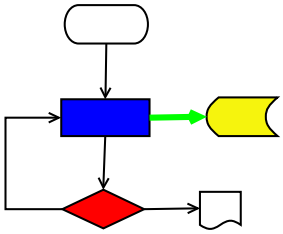
VICI



VISUAL CHART INTERPRETER

Design of vici-admin

Publication History

Date	Who	What Changes
4 October 2012	Brenton Ross	Initial version.
23 June 2014	Brenton Ross	Updated for Vici
3 November 2014	Brenton Ross	Added increment plan reference.
29 March 2016	Brenton Ross	Updated to reflect implementation.

Table of Contents

1 Introduction

This is part of the system design document for the VICI project.

1.1 Scope

This document covers the design of the vici-admin program, excluding the libraries libebnf and libsyntax which have their own documents.

This design is for increment #4.

1.2 Overview

The detailed design includes:

- Interface Stubs: A framework of facade classes for the modules.
- Use Case Descriptions: A description of how a user is expected to interact with the application.
- Application Design: The classes and their relationships.
- User Interface Design: The design and layout of the graphical components of the system.
- Persistent Storage Design: The specifications for the XML files used to store configuration and scripts.

1.3 Audience

This document is intended to be used by the designers and developers, and later the maintainers, of the VICI project.

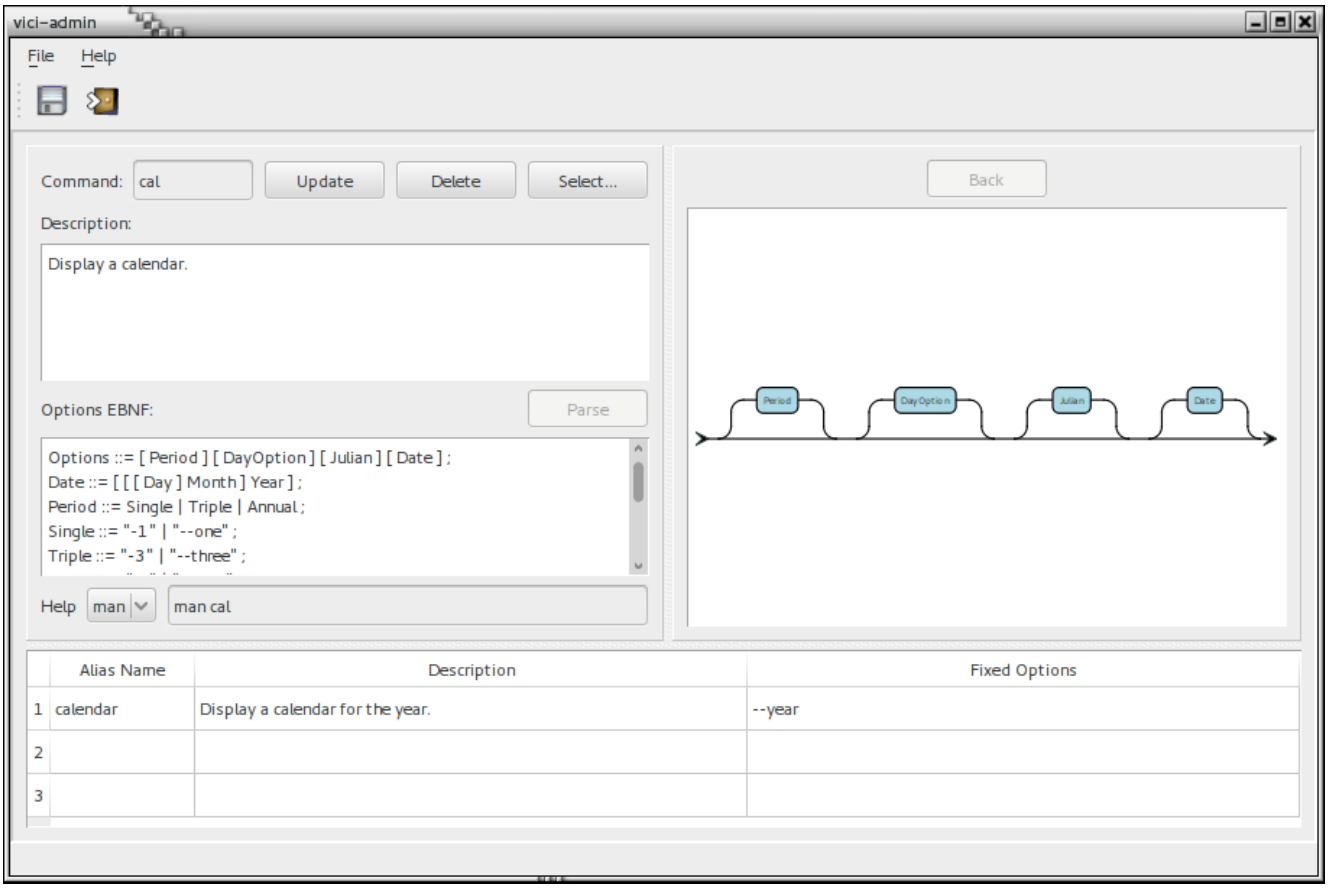
2 Responsibilities

This section lists the responsibilities that the module must implement.

T1.1	Collect the name of the command.
T1.2	Collect the syntax of the options.
T1.3	Collect a short description of the command.
T1.4	Collect the commands to show the help documentation.
T1.5	Construct an XML document from the user input.
T1.8	Restore user entered data from the XML document.
T1.11	Collect a name and some options to be used as an alias command.
T1.13	Export selected commands.
T1.14	Import commands.

3 User Interface Design

This section provides a mock up of the user interface for the module.

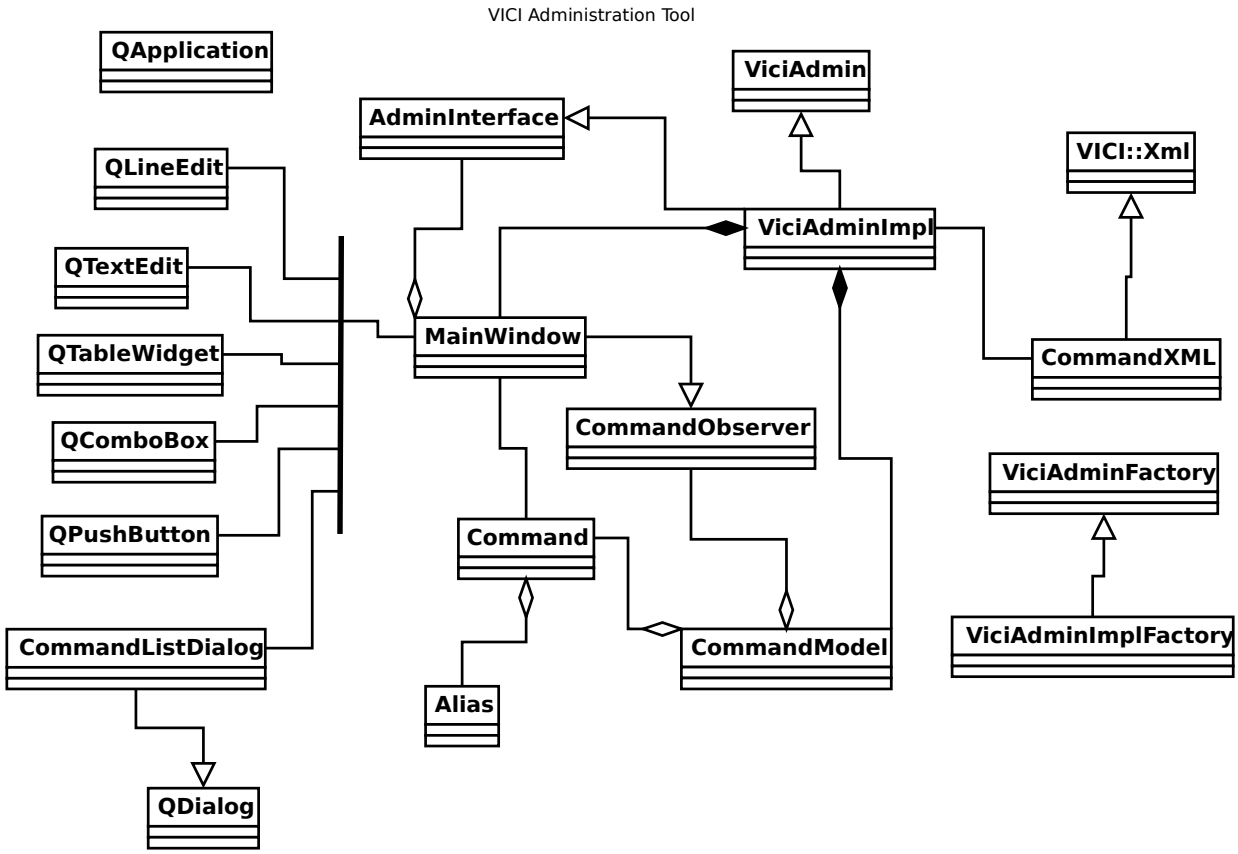


The area on the right is for the libsyntax library to display the syntax chart for the EBNF.

Please refer to the user guide for a description of how the user would interact with this program.

4 Application Design

This section shows the main classes that implement the application and their static relationships.



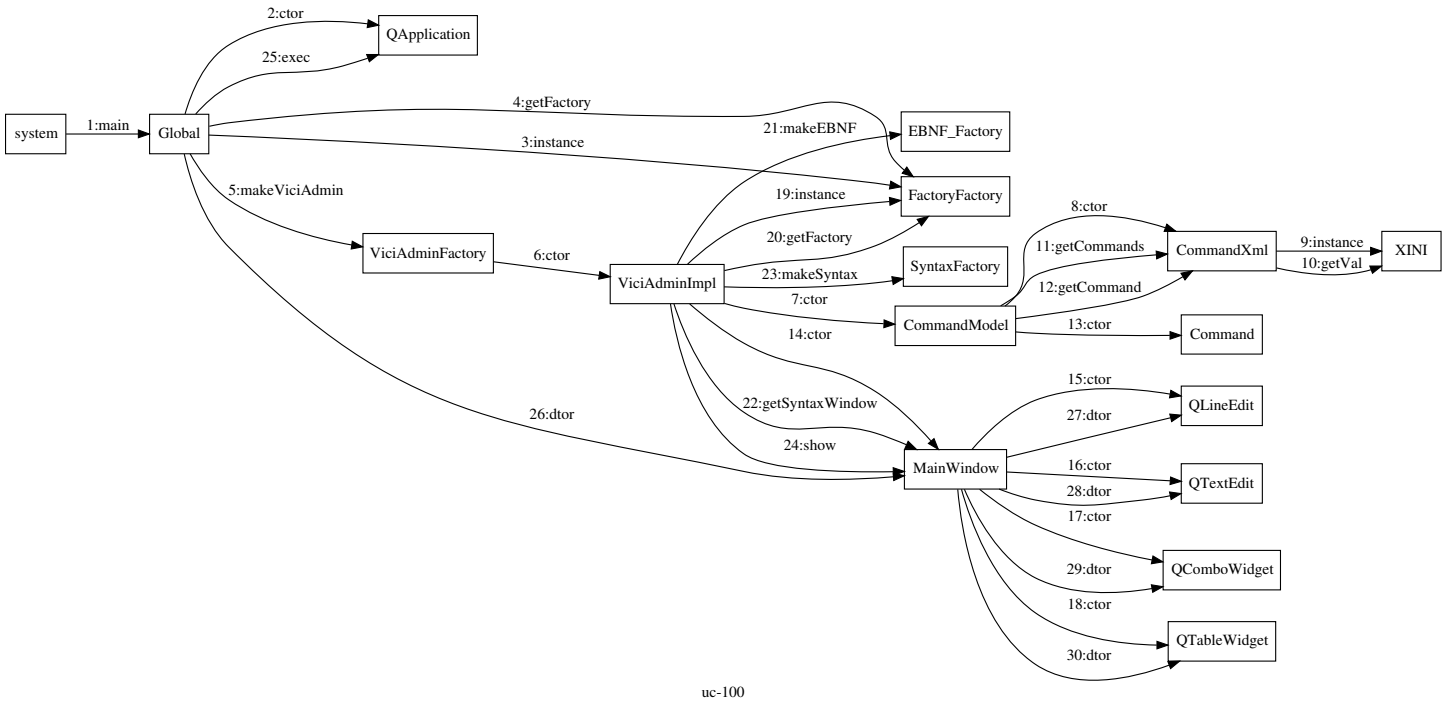
5 Collaboration Diagrams

This section documents the interactions between the classes for each applicable use case.

The edges in the diagram are numbered according to the sequence of events.

5.1 UC-100: Open the Administration Program

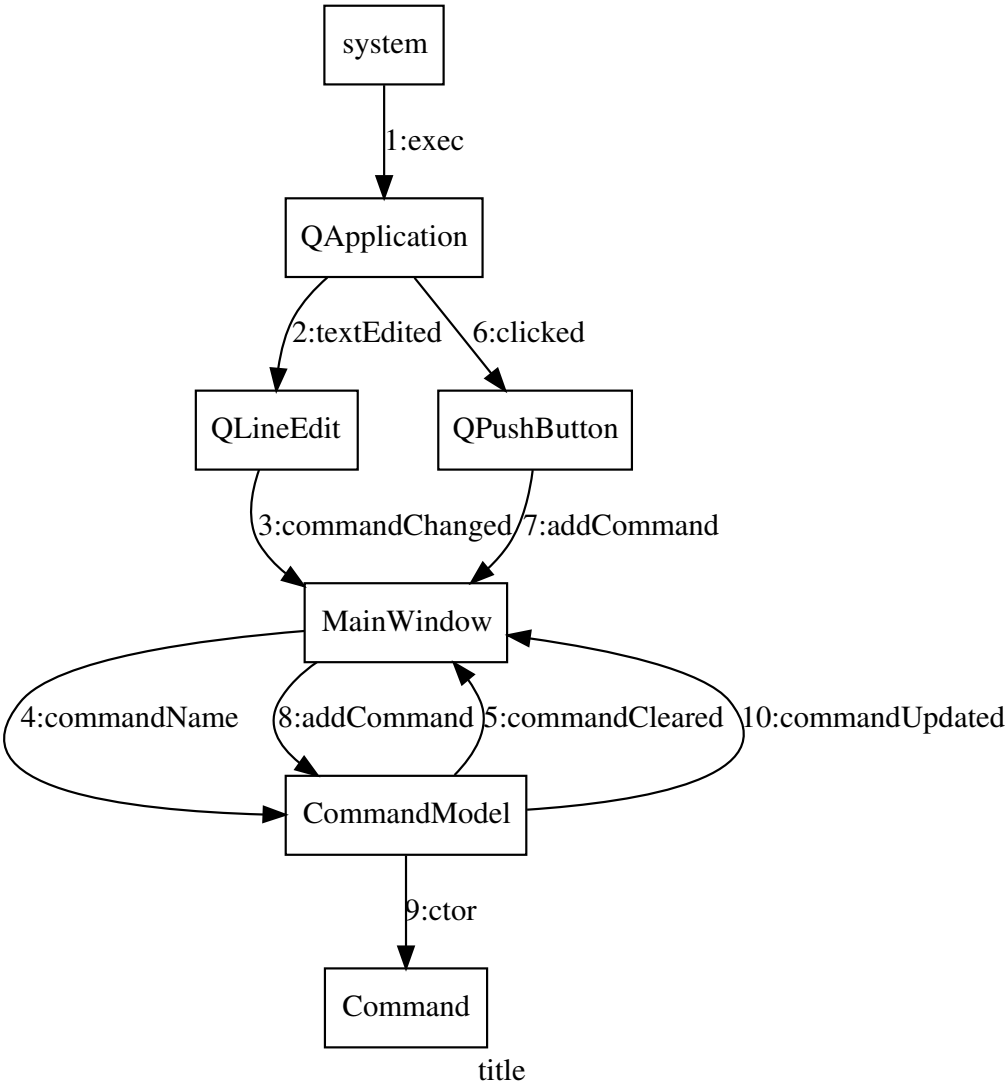
This shows the start up and shut down of the administration application.



uc-100

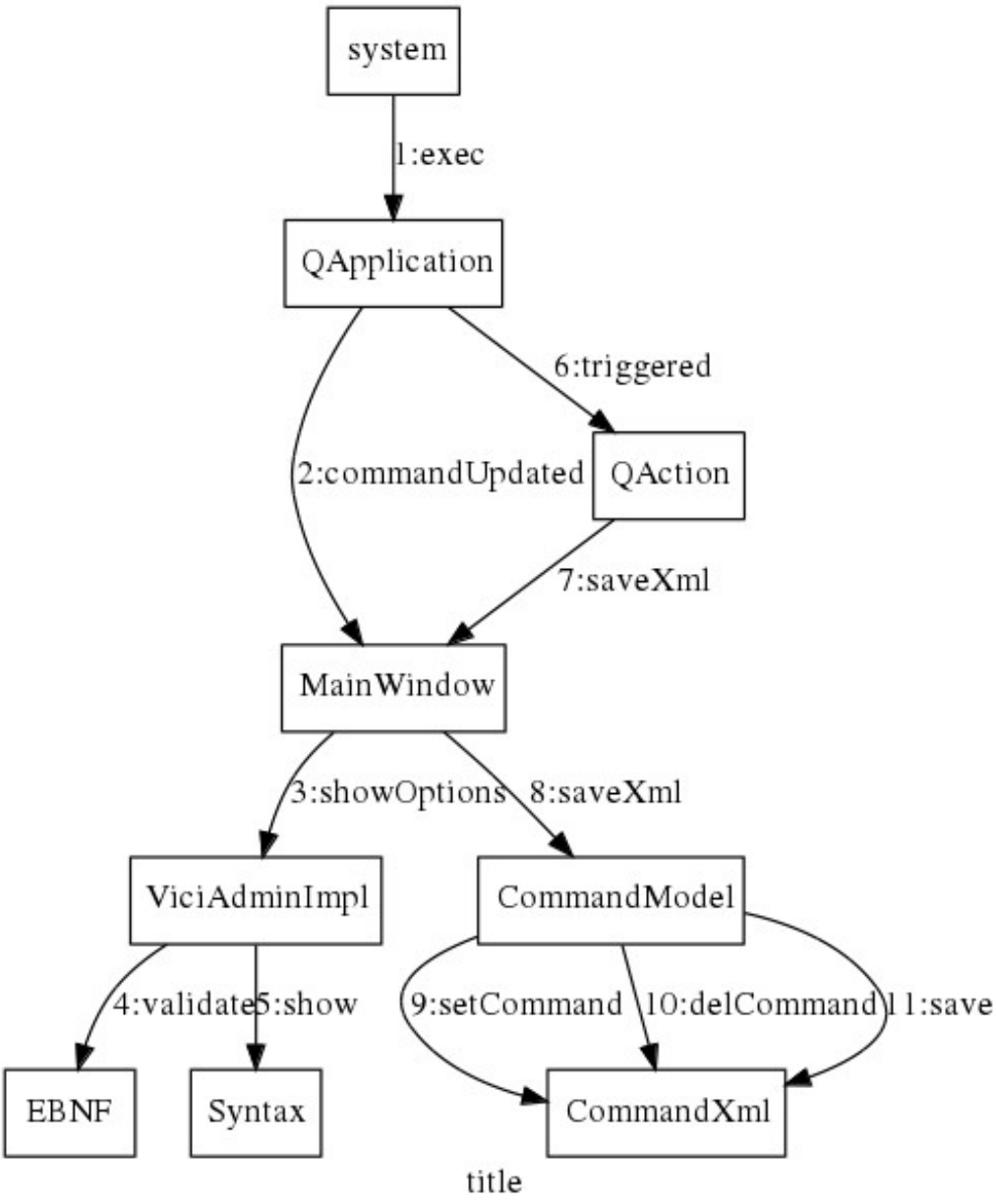
5.2 UC-101: Prepare a Command

The user enters the details of a command.



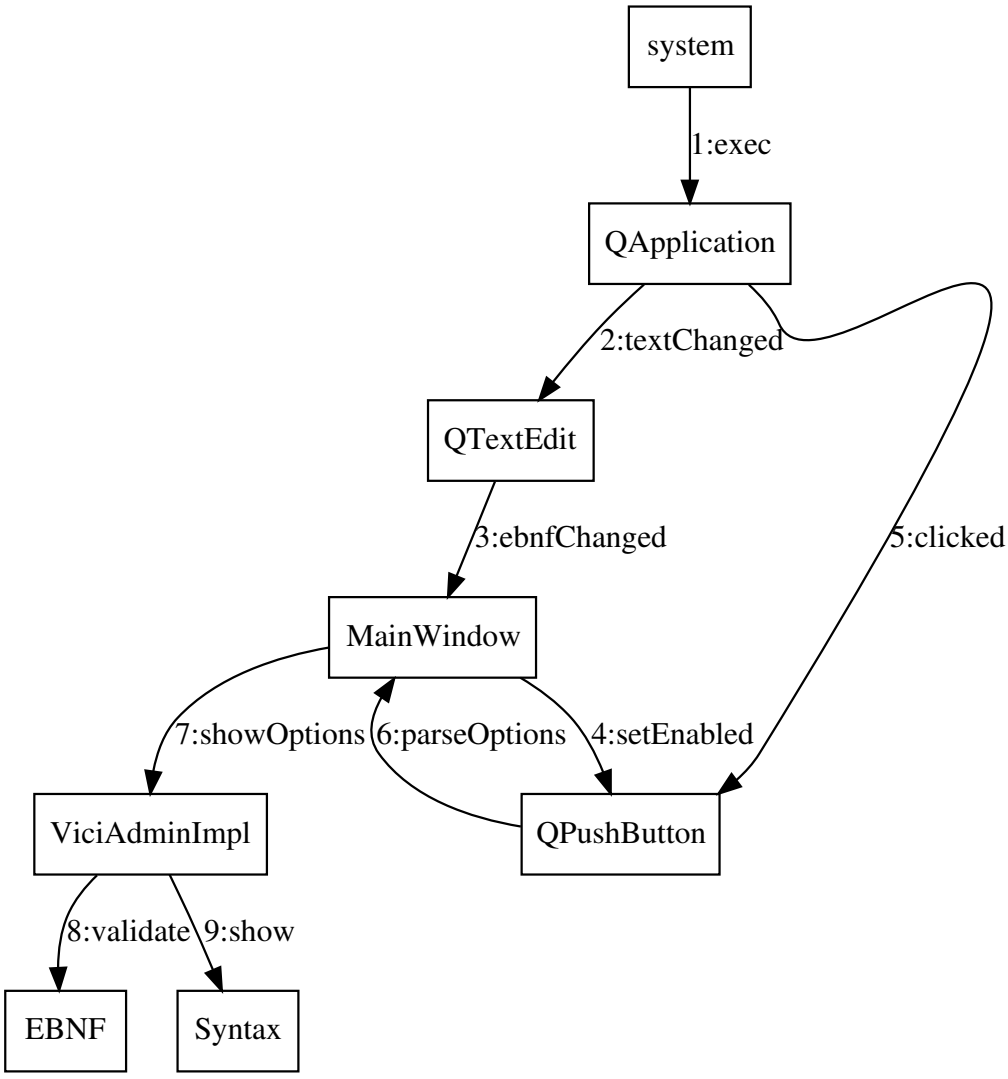
5.3 UC-102: Validate a Command

This follows on from the previous diagram with the system validating the EBNF, and the user saving the results.



5.4 UC-103: Edit a Command

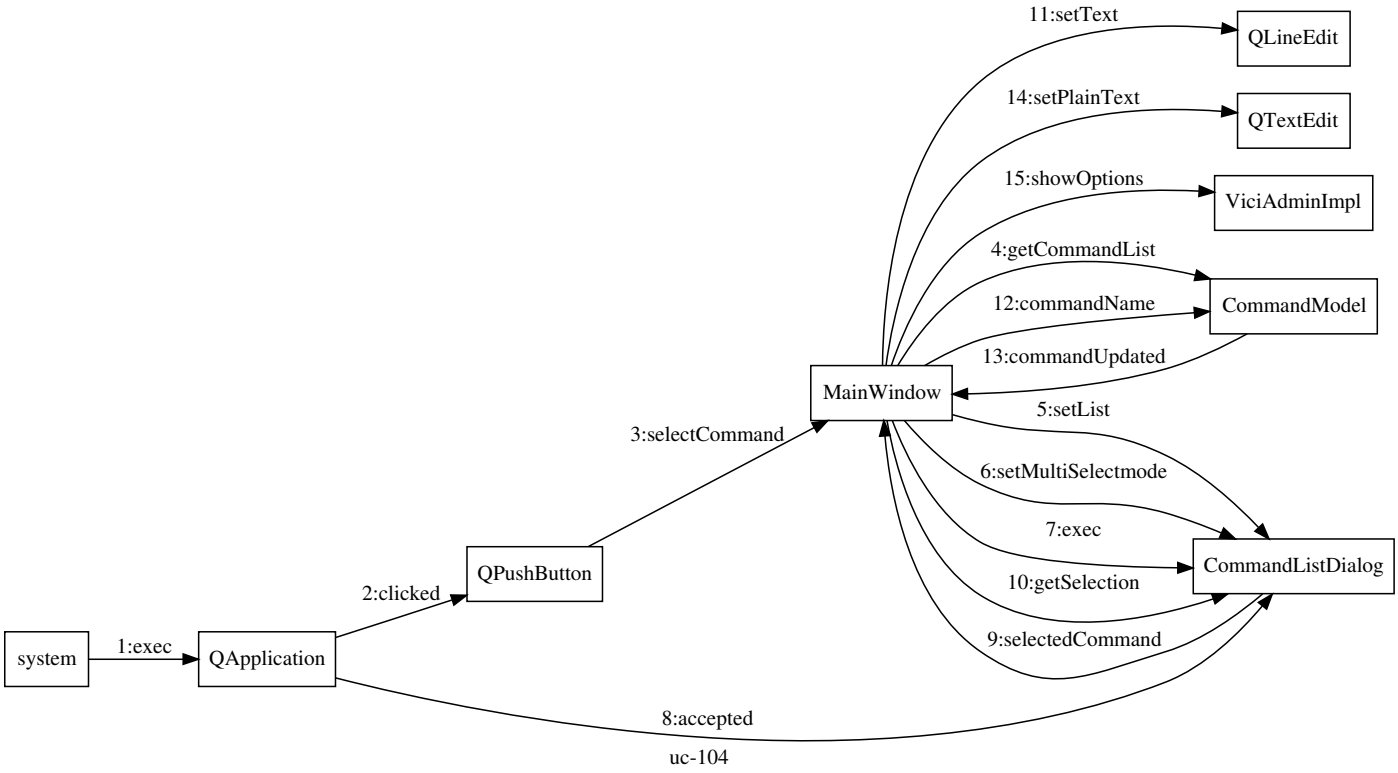
This involves editing an existing command.



title

5.5 UC-104: Select a Command

The user selects from the existing commands.



6 Class Design

This section provides our initial design for the classes. It will initially just show the major classes and their main relationships. As the design evolves these diagrams will be refined.

Please refer to the libadmin.h file for details of the class interfaces.

6.1 MainWindow Class

This class is responsible for providing the main window for the administration application.

T1.1.1	Text entry field for the name of the command.
T1.2.1	Multi-line text entry field for EBNF of the options and parameters.
T1.2.2	Provide a window for the display of a syntax chart.
T1.3.1	Multi-line text entry field for a short description of the command.
T1.4.1	Text entry fields for the help commands.
T1.4.2	Provide a window for the display of help text.
T1.11.1	Text entry field for an alias name.
T1.11.2	Text entry field for the fixed options for the alias command.

6.2 Command Class

This class holds the internal representation of a command for the administration program.

T1.1.2	Hold the name of the command.
T1.1.3	Verify that the command exists.
T1.2.3	Hold the EBNF string for the options and parameters.
T1.2.4	Cause the EBNF to be parsed.
T1.2.5	Hold the status of the command.
T1.3.2	Hold the short description of the command.
T1.4.3	Hold the commands for getting help text.
T1.11.3	Hold alias and fixed options.

6.3 CommandModel Class

This manages the internal representation of the data for the program. It contains an instance of the Command class that mirrors to MainWindow, plus a list containing a Command object for each command.

T1.5.1	Save self using CommandXML
T1.8.1	Restore self using CommandXML

6.4 CommandXML Class

This is a specialisation of the XML_Wrapper class for the Command data.

T1.5.2	Save an XML file.
T1.8.2	Load an XML file.

6.5 XML_Wrapper Class

This class provides a C++ interface to the libxml2 library. (It needs to be in a separate library as its a component used by all parts of GUISH.)

T1.5.3	Create an XML document.
T1.5.4	Create elements in the document.
T1.5.5	Create content for elements in a document.
T1.5.6	Create attributes for elements of a document.
T1.6.1	Save XML document to disk.
T1.7.1	Load an XML document from disk.
T1.8.3	Retrieve elements from a document.
T1.8.4	Retrieve content from a document.
T1.8.5	Retrieve attributes from a document.

Appendix A