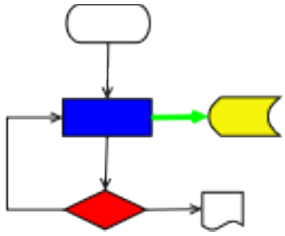
VICI



VISUAL CHART INTERPRETER

Interface Design

Publication History

Date	Who	What Changes
4 October 2012	Brenton Ross	Initial version.
21 May 2014	Brenton Ross	Updated to VICI

Table of Contents

1 Introduction

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

1.1 Scope

This document covers the interfaces between the component modules of the system.

1.2 Overview

The detailed design includes:

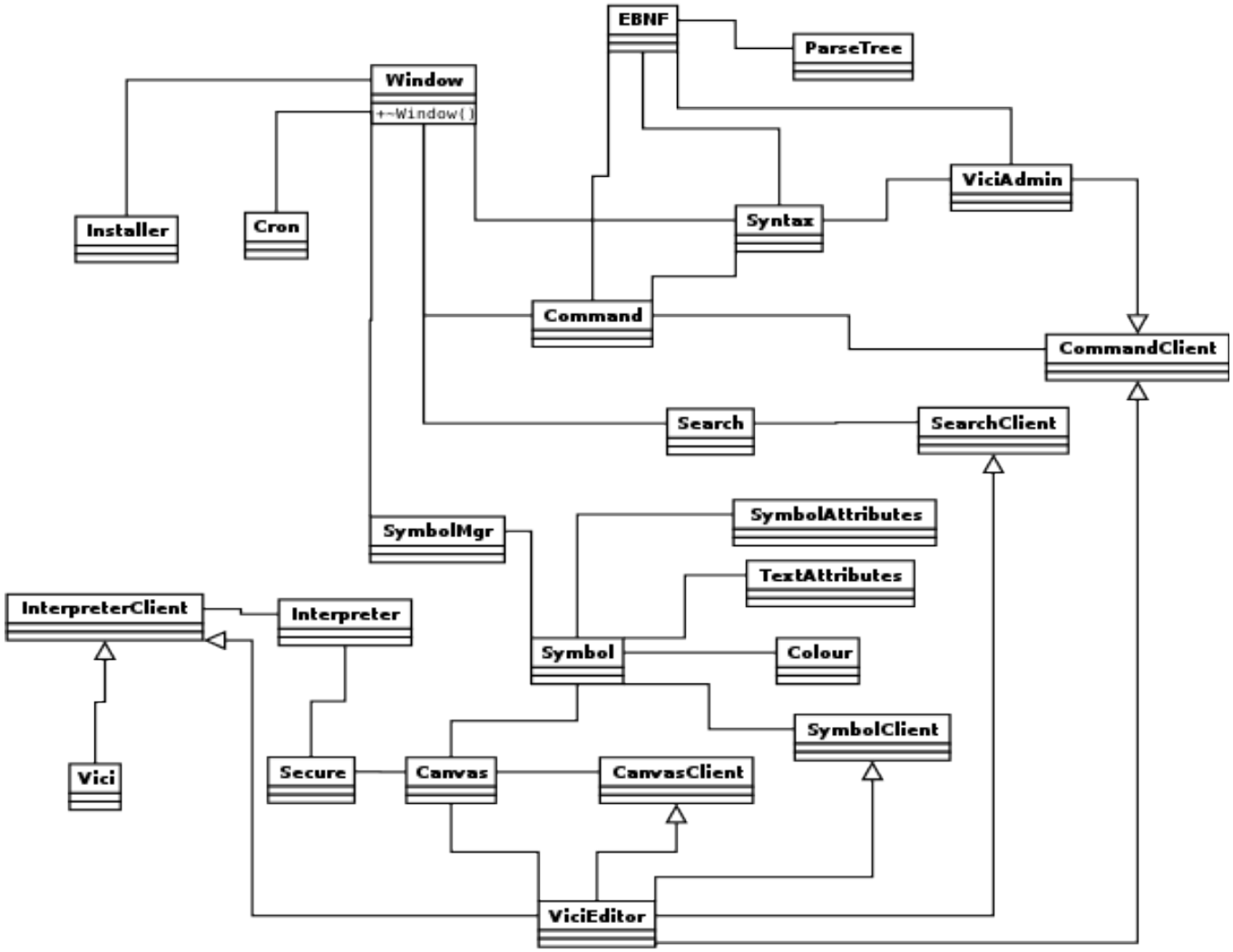
- Interface Stubs: A framework of facade classes for the modules. (This document.)
- Use Case Descriptions: A description of how a user is expected to interact with the application.
- Application Design: The classes and their relationships.
- User Interface Design: The design and layout of the graphical 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 Interface Stubs

The interface stubs provide the scaffolding that allows us to build the modules independently of each other (if that is a requirement). It provides the API for each module, and can support test and monitoring code that can be used to do module level tests.



2.1 Interface Classes

These classes are the API interfaces for each module of the design. They exist in the VICI namespace and are defined in vici.h.

2.1.1 Window

The Window class provides a strongly typed abstract pointer to window objects. This keeps the implementation details out of the global name space, while allowing references to windows to be passed between modules.

2.1.2 ParseTree

The ParseTree class provides a strongly typed pointer to a tree. This keeps the implementation details out of the global name space, while allowing the structure to be passed between modules.

2.1.3 EBNF

The EBNF class is a facade for libebnf. It provides the following public methods:

Methods	
validate	Confirm that an EBNF definition is valid.
getError	Get location and nature of last detected error.
parse	Creates a ParseTree from a string containing the EBNF definition for a command's options and parameters.

2.1.4 Syntax

The Syntax class is a facade for libsyntax. It provides the following public interface:

Constructor Parameters	
Window *	The window into which it creates its interface.
EBNF *	The EBNF parser.
Methods	
show	Takes an EBNF string, uses EBNF to create a ParseTree and then generates a diagram in the provided Window.

2.1.5 ViciAdmin

The ViciAdmin class manages the administration application.

Data	
Syntax *	Used to create the syntax charts
EBNF *	Used to parse and validate the EBNF for commands.

2.1.6 SearchClient

The SearchClient class is an abstract interface that is implemented by classes that need to be notified of events generated from the search library. The following methods are declared:

Methods	
selectedCommand	This method passes the name of the command that the user has selected to the implementing class.

2.1.7 Search

The Search class is the facade for the libsearch library. It provides the following public methods:

Constructor Parameters	
SearchClient *	The owner that should receive notifications of selections.
Window *	Where to display the search interface.
Methods	
show	Displays the search interface in the provided window.

2.1.8 CommandClient

The CommandClient class is an abstract interface that the Command class uses to notify its owners of command events. It provides the following methods, which must be implemented by a derived class:

Methods	
options	Passes a string containing the options and parameters for a command.

2.1.9 Command

The Command class is the facade for the libcommand library. It provides the following public methods:

Constructor Parameters	
Window *	The window into which the module displays its interface.
CommandClient *	The object that needs to be informed of the user's selection of options and parameters.
Methods	
show	Makes the interface visible.
setCommand	Advises the module of the command that is to have its options and parameters set.

2.1.10 SymbolAttributes

The SymbolAttributes class provides a strongly typed pointer to an object that contains the attributes of a symbol. These include shape, colour, texture and patterns.

Methods	
lineColour	Returns the colour of lines.
fillColour	Returns the colour used to fill boxes.
linePattern	Returns the pattern used for lines.
fillPattern	Returns the patterns or texture used to fill boxes etc.

2.1.11 TextAttributes

The TextAttributes class provides a strongly typed pointer to an object which contains the attributes of a text object. These include font name, font weight and font size.

Methods	
fontName	Returns the name of the font.
fontWeight	Returns the boldness of the font.
fontSize	Returns the size, in points, of the font.

2.1.12 Colour

The Colour class is a redefinition of the long type that is used to specify colours.

2.1.13 Symbol

The Symbol class provides a facade for an implementation object that handles the rendering of the various symbols.

Methods	
setAttributes	Sets the symbols and text attributes that the symbol will use to render itself.
draw	Causes the symbol to rendered on the provided Canvas at the provided coordinates, with the provided scale.

2.1.14 SymbolClient

The SymbolClient class provides an abstract interface that SymbolMgr uses to notify owning classes of events generated within the symbol library.

Methods	
selection	Passes the selected symbol to the client.
symbolAttr	Passes the set of symbol attributes to the client.
textAttr	Passes the set of text attributes to the client.

2.1.15 SymbolMgr

The SymbolMgr class provides a facade for the libsymbol library.

Constructor Parameters	
Window *	The window in which the module should display itself.
Methods	
addClient	Advises of a new client that must be notified of changes.
getDefaultAttr	Get the current default symbol attributes.
getTextDefaultAttr	Get the current default text attributes.

2.1.16 Canvas

The Canvas class provides a facade for the libcanvas library.

Constructor Parameters	
Window *	The window in which the module should display itself.
CanvasClient *	The owner that is to be notified of events in this module.
Methods	
load	Read in a script and the associated diagram.
save	Write out the diagram as a script.
setExecution	Set the execution cursor on the diagram.
setCommand	Set the command for the current symbol.

2.1.17 CanvasClient

The CanvasClient class provides an abstract interface that libcanvas uses to identify owners of events generated within the library.

Methods	
newSymbol	Advises that a symbol has been placed on the canvas.

2.1.18 InterpreterClient

The InterpreterClient class provides an abstract interface that libvici uses to identify owners of events generated within the library.

Methods	
setValue	A variable has changed value.
setFile	A file has changed state.
setCursor	The execution cursor has moved.
done	Indicates that the interpreter is now waiting for further input.

2.1.19 Interpreter

The Interpreter class provides a facade for the libvici library.

Constructor Parameters	
Secure *	A pointer to the library that confirms if a script may be opened.
InterpreterClient *	A pointer to the class that must be notified of events within the Interpreter.
Methods	
setScript	Set the script to execute.
setValue	While testing, the user has modified the value of a variable.
setBreak	Set or clear a break point.
run	Begin execution of the script.
pause	Suspend execution of a script.
kill	Terminate the script.
setPosn	Set the position to resume running.
saveSnapshot	Save the current state.
loadSnapshot	Reload the current state.

2.1.20 Secure

The Secure class provides a facade for the libsecure library.

Methods	
addSignature	Add a signature to a script
verifySignature	Verify that the script has a valid signature.

2.1.21 Cron

The Cron class provides a facade for the libcron library.

Constructor Parameters	
Window *	The window in which the module should display itself.
Methods	
show	Cause the window to display.
setCurrentFile	Sets the current file as the default one to install.

2.1.22 Installer

The Installer class provides a facade for the libinstaller library.

Constructor Parameters	
Window *	The window in which the module should display itself.
Methods	
show	Cause the window to display.
setCurrentFile	Sets the current file as the default one to install.

2.1.23 ViciEditor

The ViciEditor class manages the editor functions for the application.

2.1.24 Vici

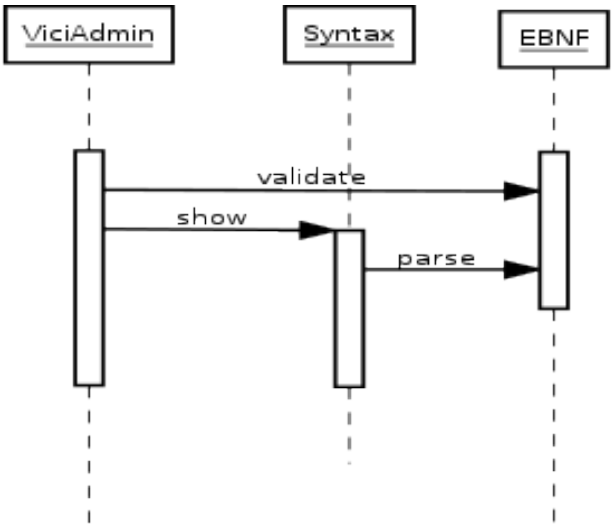
The Vici class provides a user interface for the libvici library.

3 Module Sequence Diagrams

This section provides a sequence diagram for the use cases showing how the modules interact. Any use case not shown here only uses a single module.

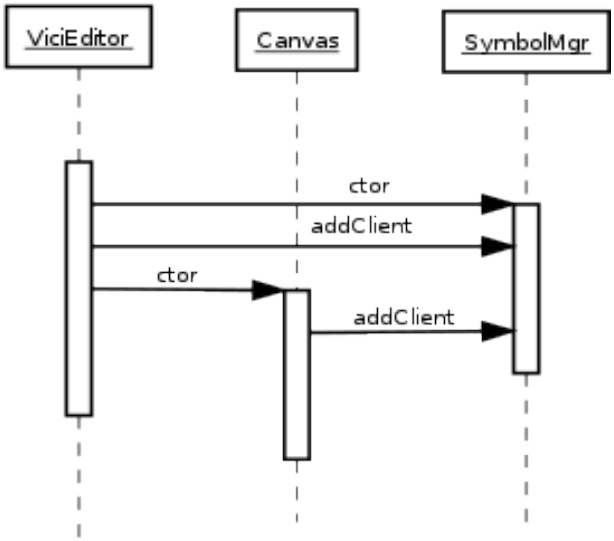
3.1 UC-102 Validate a Command

The ViciAdmin module uses the Syntax and EBNF modules to validate the EBNF syntax and to show the resulting syntax chart.



3.2 UC-106 Starting the Editor

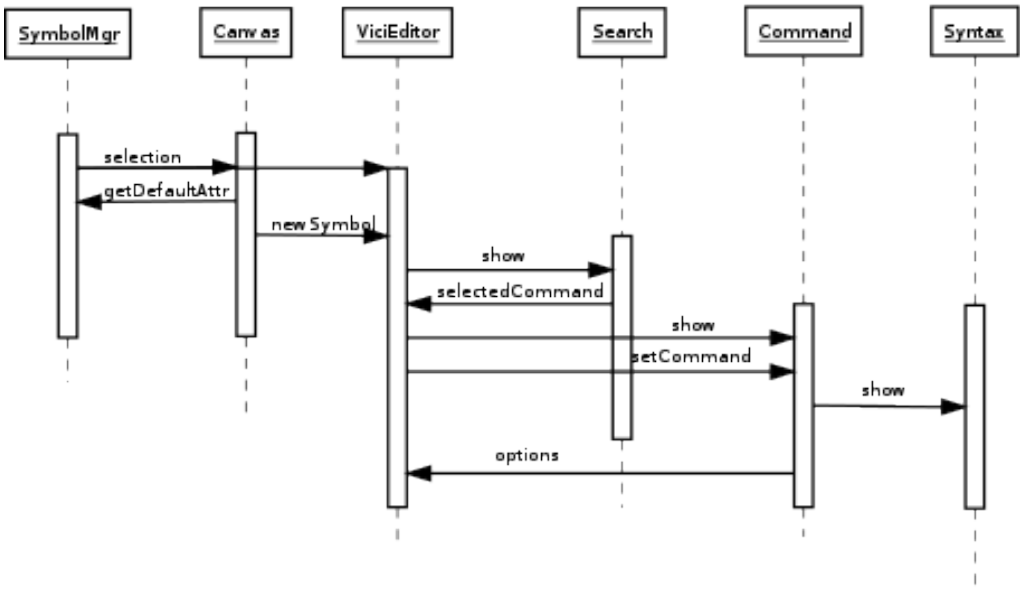
On start-up the ViciEditor instantiates a SymbolMgr and adds itself as a client to it. The ViciEditor then instantiates a Canvas to do the diagrams on and the Canvas also adds itself as a client to the SymbolMgr.



3.3 UC-107 Single Command Script

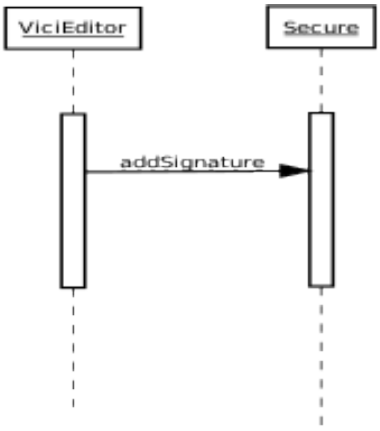
When the user selects a symbol from the palette the SymbolMgr signals the Canvas and ViciEditor of the change. The Canvas gets the details from the SymbolMgr. When the user places the symbol on the Canvas it signals the GuisEditor which responds by making the Search window visible for the user to select a command.

Once a command has been selected this is passed to the Command object so that the user can set the options and parameters with the aid of the syntax diagram.



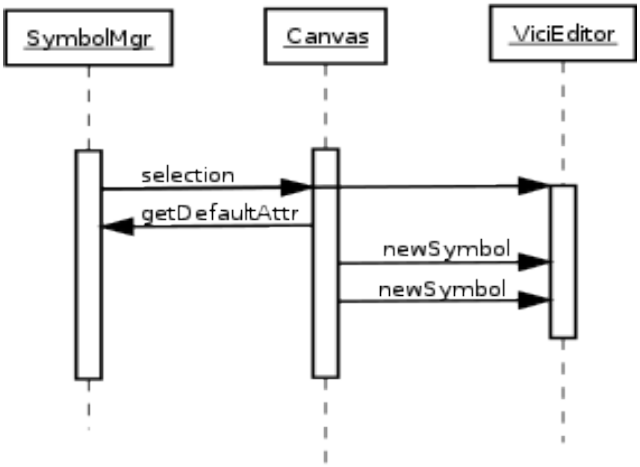
3.4 UC-108 Save a Script

When the user saves the script the ViciEditor gets the Secure object to add a signature to the script.



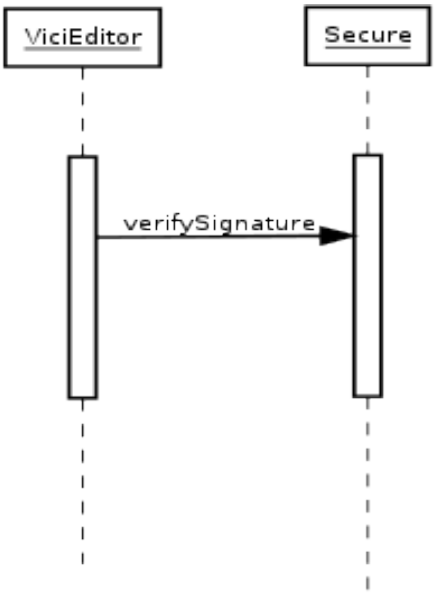
3.5 UC-109 Three Command Script

The user selects the type of connector on the symbol palette and uses it to connect command boxes.



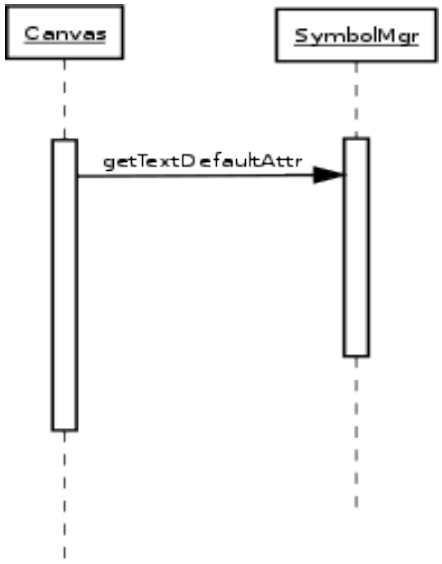
3.6 UC-113 Reload a Script

When reloading a script into the editor the system ensures that the script has a valid signature.



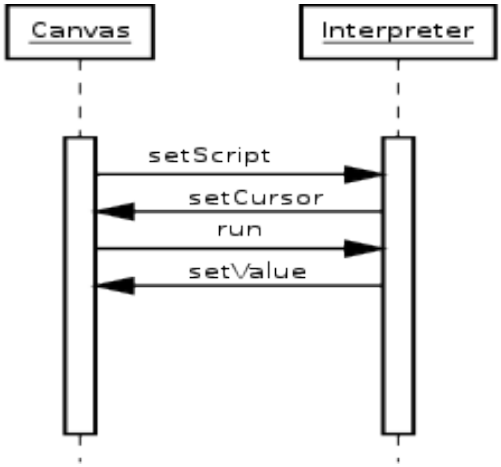
3.7 UC-115 Add Comments to a Script

The Canvas queries the SymbolMgr to get the attributes for the text being placed or modified.



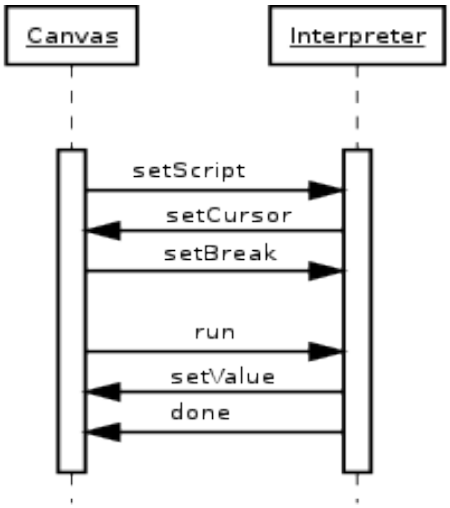
3.8 UC-137 Observe Script Operation

The Canvas advises the Interpreter of the script to run, and in return the Interpreter advises the position of the execution cursor and the values for variables.



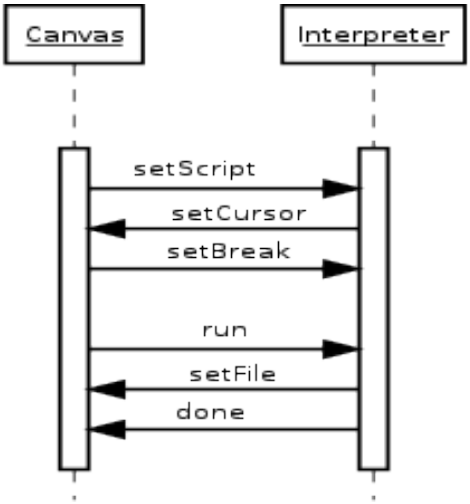
3.9 UC-138 Setting Break Points

The Canvas advises the Interpreter of the script to load. The Interpreter advises the cursor location. The script advises of the location for breakpoints, followed by the ruin command to begin execution. The Interpreter advises of the values for variables as it executes the script.



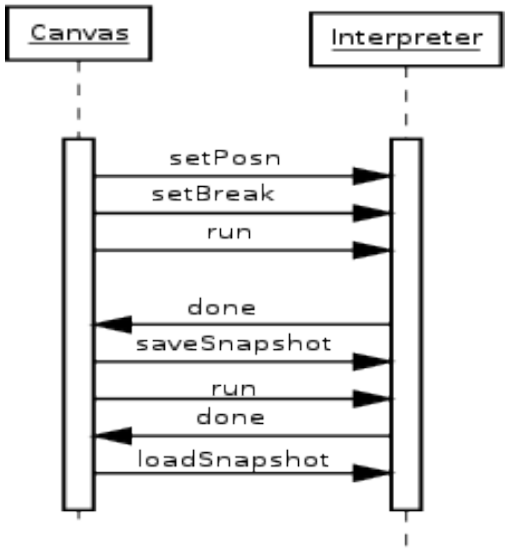
3.10 UC-140 Verify File Operations

The Canvas advises the Interpreter of the script to load. The Interpreter advises the cursor location. The script advises of the location for breakpoints, followed by the ruin command to begin execution. The Interpreter advises of the changes to files as it executes the script.



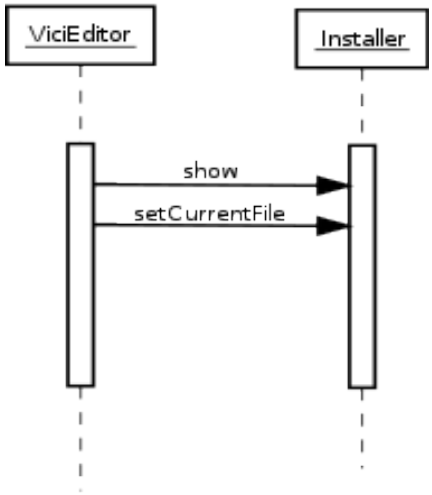
3.11 UC-141 Repeatability Tests

The user sets a break point and then saves the state in a snapshot file which is later reloaded and the tests re-done.



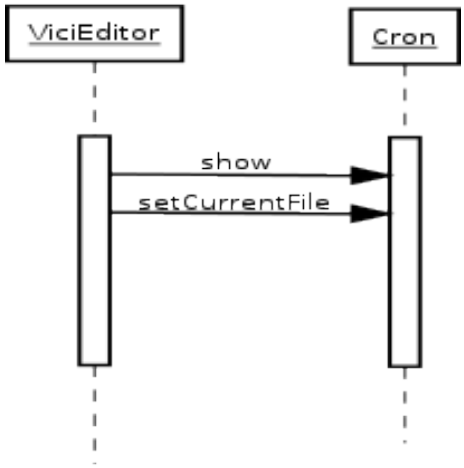
3.12 UC-142 Install a Script

The Vici Editor opens the installer window and advises it of the current script file. The installer window includes buttons to initiate the installation process, or to remove an installed script.



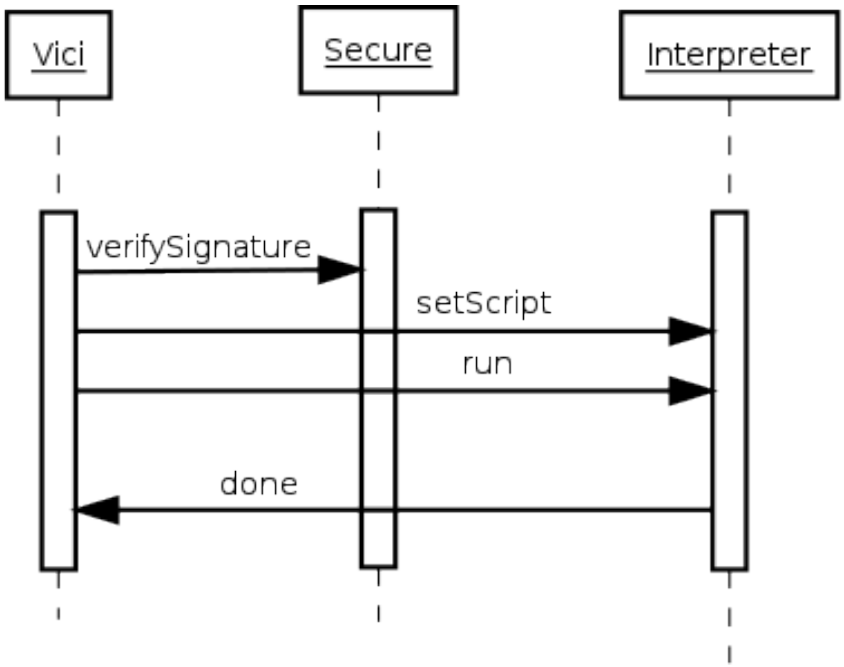
3.13 UC-144 Schedule a Script

The Vici Editor opens the scheduler window and advises it of the current script file. The scheduler window includes a user interface for scheduling a script.



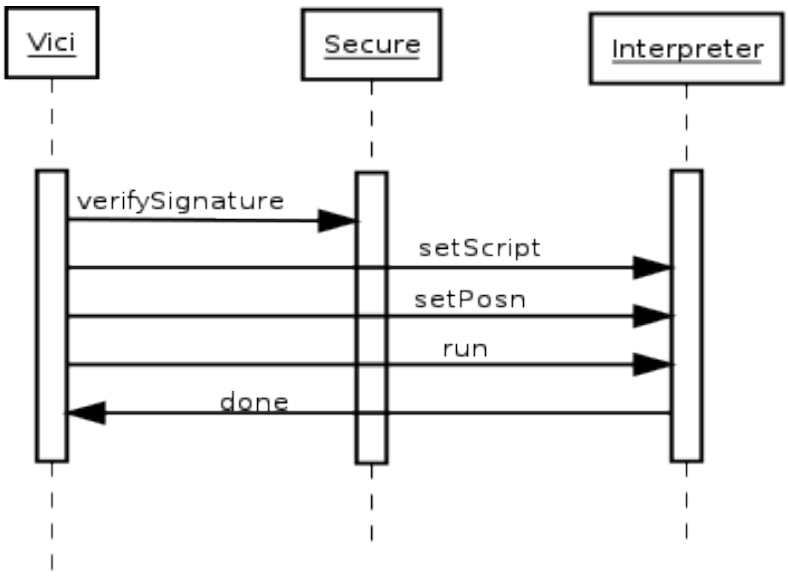
3.14 UC-145 Non-interactive Script

The Vici program verifies that the script has been correctly signed, and then waits while the Interpreter executes it.



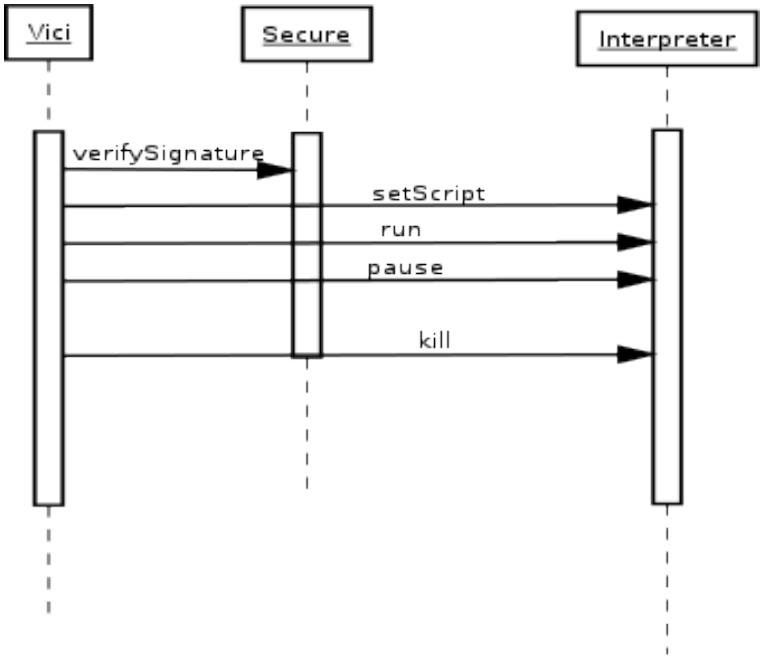
3.15 UC-148 Run a Script Function

The Vici program verifies that the script has been correctly signed then sets the starting point for execution before waiting for the script to complete.



3.16 UC-150 Control Script Operation

The Vici program provides controls that can pause continue and terminate the execution of a script. This is useful if the script starts to behave in an unexpected manner.



Appendix A