	**Titre**	User Manual for SIPInjector		
Type		User Manual	**Date**	26/04/2006
Auteur		Iassen Vassilev, Fabien Barreca	**Nombre de pages**	6
Destinataire		Bruno Dillenseger, Yoann Bersihand		

Destinataires		
Nom	**Mail**	
B. Dillenseger	bruno.dillenseger@francetelecom.com	Pour Action
Y. Bersihand	yoann.bersihand@francetelecom.com	Pour Action
F. Barreca	fbarreca@gmail.com	Pour Action
I. Vassilev	iassen_vassilev@yahoo.fr	Pour Action

Table des matières

1. Importing the plugin

1.1. General parameters

After creating a new .xis file you have to import the plugin SIPInjector by selecting the « import » tab and clicking the « Add » button.
Then you have to set the plugin's fields. These fields are :
Protocol Used by transport layer : you can choose between tcp and udp protocols.
Host's SIP User id : It's the id of a user. It's used to create the SIP URI : « *sip : USER_ID@Domain* »
Host's SIP Domain : It's the Domain used to create the sip URI (see before)
server (registrar) IP Address : It's the IP Adress of the server or the registrar you want to contact.
Your IP Adress: Your current IP address. (You can take it by doing a ipconfig on windows or ifconfig on Unix.)
Request timeout in millisec : Define the maximal time you want the injector to wait for a response to each request it sends.
Your listening port: define the port you want the sip stack to listen on.
Server's listening port: define the port the server is listening on

1.2. SDP parameters:

these parameters are crucial to make the test run.

1.2.1. Session description.

a=. Session attribute. Optional.

c=. Connection information. Optional.
Not required if included in all media.

o=. Owner/creator and session identifier.

s=. Session name.

v=. Protocol version.

1.2.2. Time description.

t=. Time the session is active.

Media description.

a=. Session attribute. Optional.

m=. Media name and transport address.

2. Defining a Behavior

(cf clif User Manual)

3. Using the SIP primitives.

When you write an ISAC scenario using the SIPInjector plugin, you can observe that it offers sample and controls primitives.

3.1. Sample primitives

bye : Sends a BYE request and waits for a response. Can be used atomically or in a scenario.
cancel: Sends a CANCEL request and waits for a response. Can be used atomically or in a scenario.
waitResponse: doesn't send anything but waits for a response code. Must be used in a logical scenario (after an invite or a register). (non sense when used atomically)

3.2. Control primitives

invite: Sends an INVITE request and DOES NOT wait for a response. (In order to handle the response, you must use a waitResponse alone or in a while loop).
register: Sends a REGISTER request and DOES NOT wait for a response. (In order to handle the response, you must use a waitResponse alone or in a while loop).
ack: Sends an ACK request and DOES NOT wait for a response. (Ack does not provocate a response so it's non sense to try to handle the response).
waitRequest: Similar to wait response, does not send anything but waits for a request.
respond: Sends a given response code (could be used after the handling of a request by waitRequest)

3.3. Test

SIPInjector also provides some test conditions. 7 testing the received responses and 5 testing the received requests: responseIsXXX. These conditions test if the last request WHICH WERE WAITING for a response received the response xxx. (The responses received when not waited are not written and can not be tested).
requestIsXXXX: these conditions test if the last request received (when waited) corresponds to XXXX. (Request, which arrived outside a waitRequest primitive, are not written and can not be tested.

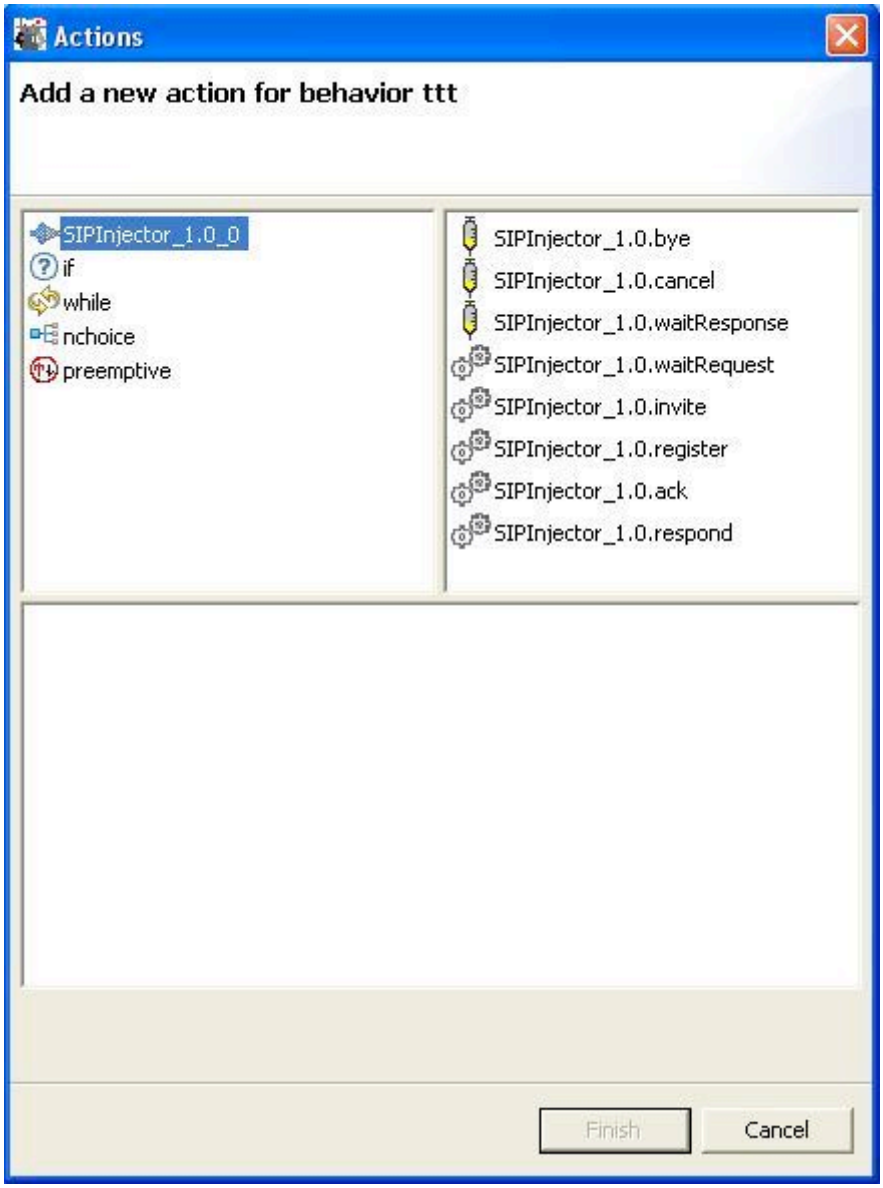

Figure 1: primitives offered by the plugin

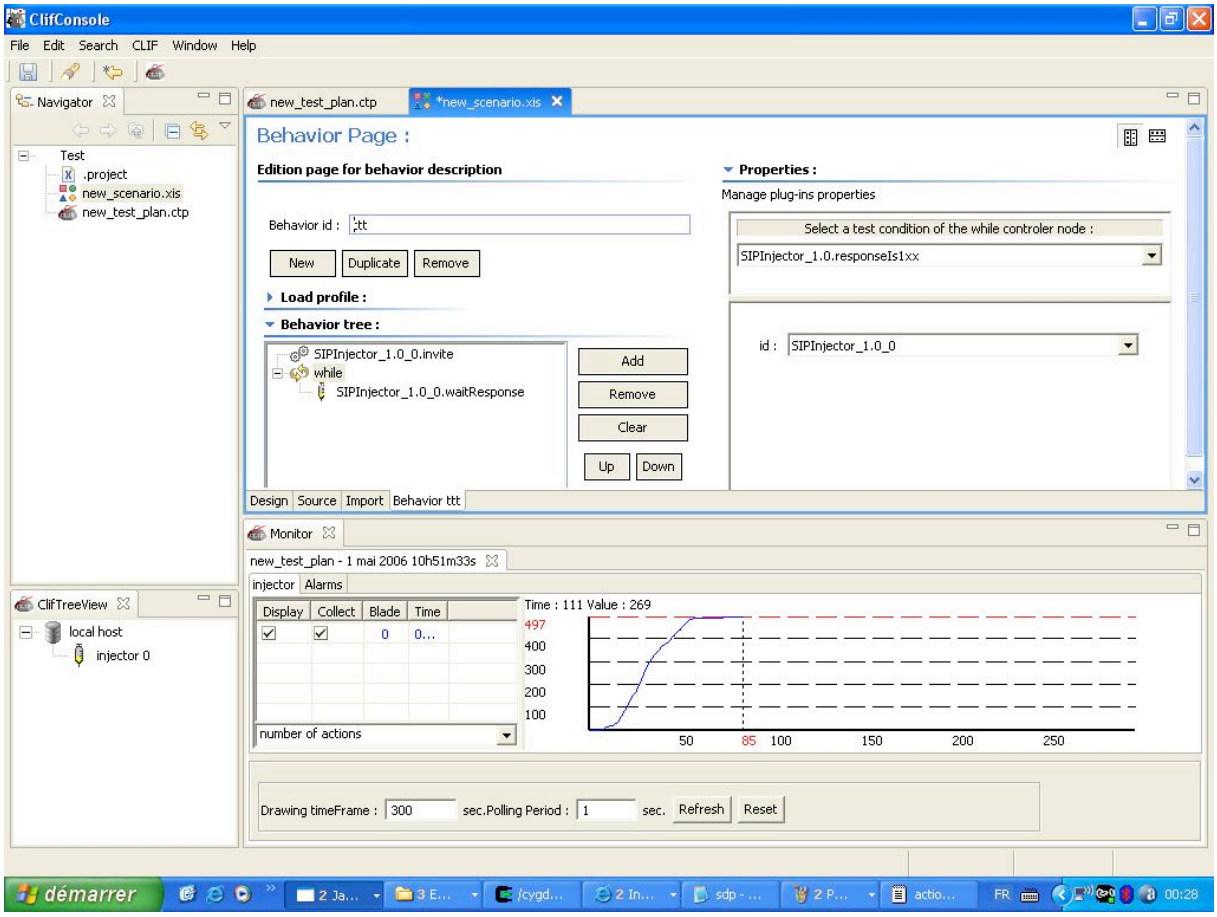

Figure 2: a classical INVITE 1XX responses handling scenario