SALT Interactors – Programming Reference

This document describes how to execute the SALT interactors in your own documents. The SALT interactors consist of specifically configured SALT markup tags, and JavaScript procedures for flow of control. For each interactor, we provide instructions for:

- Including the required SALT markup in your document
- Including the JavaScript file for interactor flow-of-control
- Executing the interactor

A Note on Using MSIE

To use these interactors with Microsoft Internet Explorer, you will need to download and install either the Microsoft Speech SDK version 1.0, or the Microsoft Speech Plug-in for Internet Explorer. Once you have done this, you will need to tell Internet Explorer to load the Speech library using an <object> tag in each document that contains SALT markup. The <object> tags for the Speech SDK and the Plug-in are different. To include the Speech SDK, use the following <object> command:

```
<object id="saltobject9677795" CLASSID="clsid:DCF68E5B-84A1-4047-98A4-0A72276D19CC"
VIEWASTEXT WIDTH=0 HEIGHT=0></object>
<?import namespace="salt" implementation="#saltobject9677795" />
```

To use the Internet Explorer Speech Plug-in, use this command:

```
<object id="saltobject9677795" CLASSID="clsid: 33cbfc53-a7de-491a-90f3-0e782a7e347a"
VIEWASTEXT WIDTH=0 HEIGHT=0></object>
<?import namespace="salt" implementation="#saltobject9677795" />
```

If you do not do this successfully, you will notice that the information inside your SALT tags is written to the screen as text.

Interactor 01 – Single List Selection

The Single List Selection Interactor allows the user to make a single selection from a limited, predefined list of choices, similar to a drop-down menu in a traditional dialog.

The interactor reads each item in the list and prompts the user to speak after an item is read to select the item.

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: li1_salt.txt

JavaScript functions: list_interactor1.js

Required Grammar Files: yesGrammar.grxml

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document
- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src="list_interactor1.js">
</script>
```

To invoke the interactor, use the following command:

```
LI1_Start(initialPrompt: string, items: string, itemGrammar: string, targetID:
string, debug: boolean, onExit: string)
```

initialPrompt: The prompt that is read at the start of the interactor

items: The list of options to choose from, separated by commas (***no spaces***)

itemGrammar: A URL to an SRGS grammar file containing the items in the list

targetID: The ID of the field to receive the result

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra
- elapsedTime
- match
- noMatch
- noResponse
- error

onExit: The function to run when the interactor completes

Interactor 02 – Multiple List Selection

The Multiple List Selection Interactor allows the user to select one or more items from a short, fixed list of selections, similar to a multiple selection box in a standard graphical interface.

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: li2_salt.txt

JavaScript functions: list_interactor2.js

Required Grammar Files: yesGrammar.grxml

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document
- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src="list_interactor2.js">
</script>
```

To invoke the interactor, use the following command:

```
LI2_Start(initialPrompt: string, continuePrompt: string, items: string, itemGrammar:
string, targetID: string, debug: boolean, onExit: string)
```

initialPrompt: The prompt that is read at the start of the interactor

continuePrompt: The prompt read after a user selection (unused)

items: The list of options to choose from, separated by commas (***no spaces***)

itemGrammar: A URL to an SRGS grammar file containing the items in the list

targetID: The ID of the field to receive the result

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra
- elapsedTime
- match
- noMatch
- noResponse
- error

onExit: The function to run when the interactor completes

Interactor 02A – Help List Selector

The Help List Interactor allows the user to simply say the name of each desired item, and indicate when they have completed their selections. If the user speaks a name that is not

in the list, or asks for help, the interactor reads the list as in the normal Multiple List Selector.

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: li2a_salt.txt

JavaScript functions: list_interactor2a.js

Required Grammar Files: yesGrammar.grxml, helpGrammar.grxml

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document
- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src="list_interactor2a.js">
</script>
```

To invoke the interactor, use the following command:

```
LI2a_Start(initialPrompt: string, listPrompt: string, items: string, itemGrammar:
string, targetID: string, debug: boolean, onExit: string)
```

initialPrompt: The prompt that is read at the start of the interactor

listPrompt: The prompt that is read if the user asks for help

items: The list of options to choose from, separated by commas (***no spaces***)

itemGrammar: A URL to an SRGS grammar file containing the items in the list

targetID: The ID of the field to receive the result

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra
- elapsedTime
- match
- noMatch
- noResponse
- error

onExit: The function to run when the interactor completes

Interactor 03 – Multiple List Selection with Deselection

The Multiple List Selection with De-selection interactor allows the user to select one or more items from a short, fixed list of choices, similar to a multiple selection box in a standard graphical interface. The user is prompted to deselect any "extra' selections at the end of the interactor.

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: li3_salt.txt

JavaScript functions: list_interactor3.js

Required Grammar Files: yesGrammar.grxml

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document
- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src="list_interactor3.js">
</script>
```

To invoke the interactor, use the following command:

```
LI3_Start(initialPrompt: string, continuePrompt: string, deselectPrompt: string,
items: string, itemGrammar: string, targetID: string, debug: boolean, onExit:
string)
```

initialPrompt: The prompt that is read at the start of the interactor

continuePrompt: The prompt read after a user selection

deselectPrompt: The prompt read just before deselection begins

items: The list of options to choose from, separated by commas (***no spaces***)

itemGrammar: A URL to an SRGS grammar file containing the items in the list

targetID: The ID of the field to receive the result

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra

- elapsedTime
- match
- noMatch
- noResponse
- error

onExit: The function to run when the interactor completes

Interactor 04 – Double Yes/No

When asked yes/no question, the user says something else (yeah, may be, neh …). The interactor will confirm an unclear response by asking, "Did you say Yes/No?" Will be used on error (user says something but yes/no) or on low confidence level (user says something close to yes/no).

Computer asks "Would you like fries with your meal?"

User responds, "Yes"

If confidence is less than a threshold then the computer asks, "Did you say yes?"

User responds, "Yes"

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: dyn4_salt.txt

JavaScript functions: dyn4_interactor.js

Required Grammar Files: dynYesNo.grxml

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document
- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src="dyn4_interactor.js">
</script>
```

To invoke the interactor, use the following command:

```
DYN4_Start(initialPrompt: string, yesPrompt: string, noPrompt: string, targetID:
string, debug: boolean, onExit: string, confirmPrompt)
```

initialPrompt: The prompt that is read at the start of the interactor

yesPrompt: The prompt that is read if the speech engine recognizes a "yes" response with low confidence

noPrompt: The prompt that is read if the speech engine recognizes a "no" response with low confidence

targetID: The ID of the field to receive the result

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra
- elapsedTime
- match
- noMatch
- noResponse
- error

onExit: The function to run when the interactor completes

confirmPrompt: The subject of the yes/no question

Interactor 05 – Double Yes/No Multimodal

When asked yes/no question, the user says something else (yeah, may be, neh …), the interactor will confirm an unclear response by displaying a dialog box asking the original question. This confirmation will be used on error (user says something but yes/no) or on low confidence level (user says something close to yes/no).

Computer asks "Would you like fries with your meal?"

User responds "Yes"

If confidence is less than a threshold then the computer asks "Would you like fries with your meal?"

User responds "Yes"

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: dyn5_salt.txt

JavaScript functions: dyn5_interactor.js

Required Grammar Files: dynYesNo.grxml

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document
- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src="dyn5_interactor.js">
</script>
```

To invoke the interactor, use the following command:

```
DYN5_Start(initialPrompt: string, yesPrompt: string, noPrompt: string, targetID:
string, debug: boolean, onExit: string, confirmPrompt)
```

initialPrompt: The prompt that is read at the start of the interactor

yesPrompt: The prompt that is read if the speech engine recognizes a "yes" response with low confidence (unused)

noPrompt: The prompt that is read if the speech engine recognizes a "no" response with low confidence (unused)

targetID: The ID of the field to receive the result

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra
- elapsedTime
- match
- noMatch
- noResponse
- error

onExit: The function to run when the interactor completes

confirmPrompt: The subject of the yes/no question

Interactor 06 – N-Best List

Confirm an answer to a "textual" question using a proximity threshold in the n-best list. After user's response, get the list of words that match that response the most, and prompt the user again with that list of words: "Did you say item1, item2 or item3?"

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: nb6_salt.txt

JavaScript functions: n_best6.js

Required Grammar Files: yesGrammar.grxml

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document
- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src="n_best6.js">
</script>
```

To invoke the interactor, use the following command:

```
NB6_Start(initialPrompt: string, targetID: string, choicePrompt: string, grammar:
string, debug: boolean, returnXML: boolean, onExit: string)
```

initialPrompt: The prompt that is read at the start of the interactor

targetID: The ID of the field to receive the result

choicePrompt: The prompt read if an n-best list is returned

grammar: A URL to an SRGS grammar file containing the expected user responses. For this interactor, the grammar must be specially formatted – items that are potentially ambiguous must contain a special tag named "nbest", with a comma-separated list of similar choices.

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra
- elapsedTime
- match
- noMatch
- noResponse
- error

returnXML: True if the interactor should return an XML recognition structure instead of the text of the recognition

onExit: The function to run when the interactor completes

Interactor 07 – N-ary Prompt

Interactor 7 seeks to correct bad user responses with a set of clarifying prompt. Clarifying prompts are played (and the interactor listens again) if the user does not respond, gives an invalid response, or asks for help.

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: na7_salt.txt

JavaScript functions: n_ary7.js

Required Grammar Files: none

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document
- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src="n_ary7.js">
</script>
```

To invoke the interactor, use the following command:

```
NA7_Start(initialPrompt: string, targetID: string, helpPrompt: string,
noMatchPrompt: string, noResponsePrompt: string, grammar: string, debug: boolean,
returnXML: boolean)
```

initialPrompt: The prompt that is read at the start of the interactor

targetID: The ID of the field to receive the result

helpPrompt: The prompt to read if the user asks for help – must be backed up in the user grammar with items that return "help" as their value.

noMatchPrompt: The prompt that is played when the user response is not recognized

noResponsePrompt: The prompt that is played when the user does not respond

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra

- elapsedTime
- match
- noMatch
- noResponse
- error

returnXML: True if the interactor should return an XML recognition structure instead of the text of the recognition

Interactor 08 – Context-Sensitive Help

Prompt the user with additional cues if he gives an incorrect response – the additional cues may be different based on the type of incorrect response.

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: cs8_salt.txt

JavaScript functions: cs_help8.js

Required Grammar Files: none

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document
- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src="cs_help8.js">
</script>
```

To invoke the interactor, use the following command:

```
CS8_Start(initialPrompt: string, targetID: string, promptNames: string, promptText:
string, noMatchPrompt: string, noResponsePrompt: string, grammar: string, debug:
boolean, returnXML: boolean, onExit: string)
```

initialPrompt: The prompt that is read at the start of the interactor

targetID: The ID of the field to receive the result

promptNames: The names of each of the specific help prompts, as specified in the grammar helpprompt tags. Separated by commas.

promptText: The text corresponding to the prompt names in promptNames – separated by newline characters.

grammar: A URL to an SRGS grammar file containing the expected user responses. For this interactor, the grammar must be specially formatted – items that the user might say that are incorrect must contain a "helpprompt" tag with the name of the prompt that

should be played to direct the user toward the correct response – prompt names must correspond to the 'promptNames' parameter provided above.

noMatchPrompt: The prompt that is played when the user response is not recognized

noResponsePrompt: The prompt that is played when the user does not respond

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra
- elapsedTime
- match
- noMatch
- noResponse
- error

returnXML: True if the interactor should return an XML recognition structure instead of the text of the recognition

onExit: The function to run when the interactor completes

Interactor 09A – Whisper Prompt A

Interactor 09.A provides the user with an example prompt right after the initial question prompt is played, without any pause.

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: wp9a_salt.txt

JavaScript functions: whisper_interactorA.js

Required Grammar Files: none

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document
- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src=" whisper_interactorA.js">
</script>
```

To invoke the interactor, use the following command:

```
WP9_Start(initialPrompt: string, whisperPrompt: string, itemGrammar: string,
targetID: string, debug: boolean, onExit: string)
```

initialPrompt: The prompt that is read at the start of the interactor

whisperPrompt: The list of examples read to clarify the initial prompt

itemGrammar: A URL to an SRGS grammar file containing the expected user response

targetID: The ID of the field to receive the result

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra
- elapsedTime
- match
- noMatch
- noResponse
- error

onExit: The function to run when the interactor completes

Interactor 09B – Whisper Prompt B

Interactor 09.B provides the user with an example prompt after the initial question prompt is played, with a short wait in between.

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: wp9b_salt.txt

JavaScript functions: whisper_interactorB.js

Required Grammar Files: none

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document

- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src=" whisper_interactorB.js">
</script>
```

To invoke the interactor, use the following command:

```
WP9_Start(initialPrompt: string, whisperPrompt: string, itemGrammar: string,
targetID: string, debug: boolean, onExit: string)
```

initialPrompt: The prompt that is read at the start of the interactor

whisperPrompt: The list of examples read to clarify the initial prompt

itemGrammar: A URL to an SRGS grammar file containing the expected user response

targetID: The ID of the field to receive the result

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra
- elapsedTime
- match
- noMatch
- noResponse
- error

onExit: The function to run when the interactor completes

Interactor 11 – Timeout Adjuster

This interactor seeks to determine how much time a user needs to respond to a typical question.

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

SALT Markup: li11_salt.txt – you will need to add your own grammar to the SALT markup as indicated

JavaScript functions: list_interactor11a.js

Required Grammar Files: yesGrammar.grxml

To include this interactor in your own files, follow these steps:

- Extract the contents of the zip archive for this interactor into the directory containing your document
- Copy the contents of the SALT markup file into your document
- Include the JavaScript file into your document with a <script> command, like this:

```
<script language="JavaScript1.2" src=" list_interactor11a.js ">
</script>
```

To invoke the interactor, use the following command:

```
LI11_Start(initialPrompt: string, item: string, targetID: string, debug: boolean)
```

initialPrompt: The prompt that is read at the start of the interactor

items: The item the user should choose

targetID: The ID of the field to receive the result

debug: True if you want the interactor to write out debugging information, false otherwise. If debugging information is written, you will need to have the following fields in your document

- correct
- incorrect
- nulls
- missing
- extra
- elapsedTime
- match
- noMatch
- noResponse
- error

Interactor 12 – Multiple Related Fields

This interactor allows the user to input data into multiple input fields in any logical order. User-provided functions allow the interactor to make sense of the ambiguities that can arise from this.

To view the state transition diagrams for the interactors:

http://salt-usability.sourceforge.net/main/8other/InterStateMachines.htm

Interactor 12 is only shown as a proof of concept – the example application given in the archive shows an example using numeric dates. This interactor is quite complex, and is not suggested for any but advanced use with extensive customization.

See the “Architectural Design” and “Detailed Design” Documents for specifics on this interactor.