


SUE: Screenreader & Usability Extensions

Spezifikation der Nutzerschnittstelle

Autoren:	Nicole Anacker, Ramona Bunk, Martina Weicht, Frank Zenker	
Datum:	15.07.2008	
Revision:	244	
Status:	Entwurf	
Copyright:	Copyright © 2007, 2008 IT Science Center gGmbH unter der BSD-Lizenz	
Quelle:	https://sue.svn.sourceforge.net/svnroot/sue/docs/sue_ui-specification_de.odt	

Zusammenfassung

Dieses Dokument dient der Beschreibung der Anwendungsschnittstelle von SUE (Screenreader & Usability Extensions) speziell für Entwickler und Tester. Die aktuelle Version dieses Dokuments beschreibt die Schnittstelle für Tastatureingaben und Sprachausgaben. Die Ausgabe auf der Braillezeile und die Benutzung der Vergrößerungssoftware werden noch nicht beschrieben.

Inhalt

Zusammenfassung.....	1
1 Einführung.....	3
1.1 Elemente der Anwendungsschnittstelle (User Interface Elements).....	3
1.2 Terminologie.....	4
2 Ausgabe von Ereignissen.....	4
2.1 Start des SUE-Screenreaders.....	5
Beispiel 1. Meldung beim Starten von SUE.....	5
2.2 Aktivieren von Fenstern.....	5
Beispiel 2. Ausgaben beim Fensterwechsel.....	5
2.3 Fokuswechsel.....	5
Beispiel 3. Fokus auf einem Button.....	6
Beispiel 4. Fokus auf einem Spinbutton.....	6
Beispiel 5. Fokus auf einer Tabelle.....	6
2.4 Auswahländerungen.....	6
2.4.1 Aktivierung eines Objektes.....	6
Beispiel 6. Aktivierte Selektion auf einem Druckknopf.....	8
Beispiel 7. Aktivierte Selektion auf einen Spin-Button.....	8
Beispiel 8. Aktivierte Selektion auf eine Tabellenzelle.....	8
2.4.2 Objekte aus- und abwählen.....	8
Beispiel 9. Selektierte Objektliste.....	8
2.4.3 Text aus- und abwählen.....	8
2.5 Zustands- und Wertänderungen.....	9
2.6 Cursorbewegungen.....	9
2.7 Text bearbeiten.....	9
2.8 Screenreader beenden.....	9
2.9 Beispiel: Nautilus.....	10
Beispiel 10. Nautilus-Fenster wird geöffnet.....	10

Beispiel 11. Nautilus-Zeilennavigation.....	10
Beispiel 12. Nautilus-Ordner einklappen und ausklappen.....	11
Beispiel 13. Nautilus-Spaltennavigation.....	11
Beispiel 14. Nautilus type-ahead-Suche.....	11
3 Tastenkombinationen.....	11
3.1 Der Cursor.....	12
3.2 Der Pointer.....	12
3.3 Betrachtung der Tastatur-Schnittstelle.....	12
3.3.1 Empfohlene Tastatur-Ergonomie.....	13
3.4 Tastaturanbindung.....	14
3.5 ReviewScript-Funktionen.....	15
3.6 BasicSpeechScript-Funktionen.....	16
3.7 BookmarkScript-Funktionen.....	17
3.8 SearchScript-Funktionen.....	18
3.9 DefaultDialogScript-Funktionen.....	18
3.10 DeveloperScript-Funktionen.....	18
3.11 Anwendungsspezifische Tastenkürzel.....	19
4 Grafische Dialoge.....	19
4.1 Settings Chooser.....	19
4.1.1 Script Settings.....	19
4.1.2 Device Settings.....	20
4.1.3 System Settings.....	21
4.1.4 Profile Settings.....	21
4.1.4.1 Installing Extensions.....	21
4.1.4.2 Associating Extensions.....	22
4.2 Help Chooser.....	22
4.2.1 Control Help.....	22
4.2.2 Accessibility Info.....	23
4.2.3 Command Help.....	23
4.2.4 Application Info.....	23
4.3 Search Chooser.....	23
4.4 Script Chooser.....	23
4.5 Developer Dialogs.....	24
4.5.1 Event Monitor.....	24
4.5.2 Task Monitor.....	25
4.5.3 I/O Monitor.....	26
5 Configurable Settings.....	26
5.1 Script Settings.....	26
5.2 Device Settings.....	27
5.3 System Settings.....	29
5.4 Other Settings.....	29
Hinweis.....	29

1 Einführung

1.1 Elemente der Anwendungsschnittstelle (User Interface Elements)

SUE (Screenreader & Usability Extensions) basiert auf einer erweiterbaren Plattform für Hilfsanwendungen. Der Zugriff auf unterstützende Anwendungsprogramme und der Empfang von Ereignissen (*events*) erfolgt über die sogenannte *AccessEngine*.

Die Hauptfunktionalität von SUE wird hauptsächlich durch Scripte bereitgestellt, die verschiedene Aufgaben (*tasks*) abarbeiten, wenn entsprechende Ereignisse von der *AccessEngine*, von der Anwendung oder vom Anwender über die Tastatur ausgelöst werden. Alle Scripte, die SUE beinhaltet, bilden die Screenreader-Anwendungsschnittstelle (UI) speziell für Menschen mit Sehschädigungen.

Folgende Scripte bilden die Basis der Screenreader-Anwendungsschnittstelle:

BasicBrailleScript

Definiert Aufgaben (*tasks*) für die Interaktionen mit einer Braillezeile.

BasicMagScript

Definiert Aufgaben für die Vergrößerung der Bildschirmdarstellung.

BasicSpeechScript

Definiert Aufgaben für die Ausgabe von Ereignissen über Sprachausgabertools.

BookmarkScript

Definiert Aufgaben für die Verwendung von Lesezeichen in einer Anwendung.

DefaultDialogScript

Definiert Aufgaben für die Darstellung und Verwaltung der Konfigurationsdialoge von SUE.

DeveloperScript

Bietet zusätzliche Tastenkombinationen für die Entwicklung und Fehlersuche in Scripten.

LanguageScript

Erlaubt das Umherschalten zwischen drei Sprachen.

ReviewScript

Definiert Tastenkombinationen, die dem Anwender eine Übersicht über den Inhalt einer Anwendung geben.

SearchScript

Definiert Aufgaben (*tasks*) für die Suche von Objekten im Accessible-Baum einer Anwendung.

SUEScript

Enthält Standard-Aufgaben, die für SUE definiert sind und in kein anderes Script passen.

Weitere anwendungsspezifische Scripte korrigieren die Zugänglichkeit und die Nutzbarkeit bestimmter Anwendungen wie Metacity, GEdit und Firefox. Sie können neue Tastenkürzel und Ereignisverarbeitungen enthalten. Der Name eines Scriptes gibt an, für welche Anwendung es eingesetzt wird.

Die Screenreader-Anwendungsschnittstelle stellt eine Reihe grafischer Dialoge bereit.

Chooser sind Dialoge für komplexe Interaktionen mit dem Anwender. Sie können durch einfache Tastenkombinationen aufgerufen werden.

Folgende *Chooser* werden in SUE verwendet:

A11yChooser

SettingsChooser

Bietet Einstellungsmöglichkeiten für Scripte, Hilfsmittelgeräte, das SUE-System und Benutzerprofile.

ScriptChooser

Erlaubt das kurzzeitige manuelle Laden und Entladen von Scripten für eine bestimmte Anwendung.

SearchChooser

Definiert ein Suchdialog für das Suchen von bestimmten Objekten in der aktuellen Anwendung.

CommandChooser

Definiert einen Dialog für das Anzeigen aller aktuell registrierter Tastenkürzel in SUE.

Monitore sind Dialoge für das Protokollieren der Ereignisse in SUE und helfen bei der Fehlersuche in den Scripten.

Folgende Monitore sind in SUE enthalten:

RawEventMonitor

Zeigt die Ereignisse an, die im System auftreten und derzeit über AT-SPI zugänglich sind.

TaskMonitor

Zeigt die Aufgaben (tasks) an, die anhand von internen SUE-Events ausgelöst und in den Scripten abgearbeitet werden.

IOMonitor

Zeigt Anwendereingaben und Ausgaben von SUE, bevor sie für die Darstellung auf dem jeweiligen Gerät formatiert werden.

Der verbleibende Teil dieses Dokuments beschreibt die Funktionalität, die die beschriebenen Elemente in ihrem Zusammenspiel bieten.

Das Dokument behandelt Themen wie die Automatische Beantwortung von Ereignissen, verfügbare Tastaturkommandos, die Struktur von Dialogen und konfigurierbare Anwendereinstellungen. Jeder Bereich beschreibt die beabsichtigte, fertige Darstellung der Screenreader UI, aber gibt auch Hinweise darauf, was im aktuellen Screenreader noch fehlt.

1.2 Terminologie

Die Begriffe *sagen*, *melden*, *berichten*, *vorlesen*, usw. werden in diesem Dokument immer verwendet, wenn „SUE Informationen ausgibt“. Wie die Informationen ausgegeben werden, hängt von drei Faktoren ab.

1. Die verfügbaren Ausgabegeräte legen fest, über welche Medien die Informationen ausgegeben werden (z.B. audio, taktil).
2. Der Leistungsumfang eines verfügbaren Ausgabegerätes bestimmt, wie die Informationen ausgegeben werden (z.B. Sprach- oder Klangausgabe, Anzahl der Braille-Zeichen).
3. Konfigurationen vom Anwender für unterstützte Geräteeinstellungen legen die endgültige Ausgabe fest (z.B. Stimmlage).

Deshalb bezieht sich eine Aussage wie „SUE sagt das Label des Textfeldes an.“ nur darauf, WELCHE Informationen ausgegeben werden, nicht WIE sie ausgegeben werden.

Bei Fragen zu Begriffen aus der SUE-Architektur sollte eher das AccessEngine-Workbook herangezogen werden.

2 Ausgabe von Ereignissen

Das `BasicSpeechScript` reagiert auf eine Anzahl allgemeiner Ereignisse. Die Ausgabe, die vom `BasicSpeechScript` für jedes dieser Ereignisse erzeugt wird, wird in den folgenden Kapiteln beschrieben.

Anmerkung: Es gibt unterschiedliche Einstellungsmöglichkeiten zum Umfang der Sprachausgabe, die je nach Auswahl verschiedene Inhalte hinzufügen oder auslassen. Geräteeinstellungen und

Leistungsfähigkeit können ähnliche Effekte bewirken und die Ausgabe verändern. Die folgende Beschreibung setzt voraus, dass kein Teil der Ausgabe ausgeblendet wird (Die Wortfülle ist auf Maximum eingestellt). Außerdem wird hier nicht darauf eingegangen, wie die Ausgabe erfolgt, sondern jeweils der Inhalt angegeben, den das `BasicSpeechScript` an ein Ausgabegerät sendet.

2.1 Start des SUE-Screenreaders

Wenn SUE auf der Kommandozeile *ohne* die Option `--no-intro` gestartet wird, sagt das System:

1. Die Willkommensmeldung in der jeweilig eingestellten Landessprache mit folgendem Format:

```
Willkommen zu Screenreader Usability Extensions Version %  
(Versionsnummer), Revision %(Erstellungsdatum)
```

Wenn nach dem Start ein Fenster aktiv ist, sagt das `BasicSpeechScript`

1. „Fenster“ gefolgt vom Titel des aktiven Fensters.

Über den aktuellen Fokus oder die Cursorposition werden keine weiteren Informationen ausgegeben, bis weitere Ereignisse auftreten.

Beispiel 1. Meldung beim Starten von SUE

Der Anwender startet SUE über den Dialog „Anwendung ausführen“ (Alt+F2) und gibt im Textfeld `sue` ein. Der Texteditor `gedit` läuft als aktive Anwendung.

```
Willkommen zu Screenreader Usability Extensions Version 0.3.0, Revision  
Tue Jul 01 12:00:00 UTC 2008. Fenster Ungespeichertes Dokument 1 - gedit.
```

2.2 Aktivieren von Fenstern

Erscheint ein neues Fenster im Vordergrund, so liest das `BasicSpeechScript` folgende Informationen vor:

1. Den Titel des aktiven Fensters oder die Zeile "keine Ausgabe verfügbar".
2. Alle statischen Text-Label.
3. Die Informationen, die bei einem Fokuswechsel gemeldet werden (Siehe auch [Abschnitt 2.3, "Fokuswechsel"](#)).

Beispiel 2. Ausgaben beim Fensterwechsel

Der Anwender schließt ein geändertes `gedit`-Dokument, bevor es gespeichert wurde. Der `gedit`-Speicherbestätigungsdialog erscheint.

```
Fenster Frage Änderungen am Dokument "Ungespeichertes Dokument 1" vor dem  
Schließen speichern? Falls Sie nicht speichern, gehen in den letzten 5  
Sekunden vorgenommene Änderungen unwiderruflich verloren.
```

Die Meldungen für einen Fokuswechsel und eine aktive Selektionsänderung folgen direkt darauf.

2.3 Fokuswechsel

Ein Fokuswechsel-Ereignis weist den Anwender darauf hin, dass alle weiteren Eingaben an ein neues Objekt gerichtet werden. Wenn der Fokus zu einem neuen Objekt im aktiven Fenster wechselt, spricht das `BasicSpeechScript`:

1. Den Namen oder Label des Containers mit dem fokussierten Objekt (z.B. Label einer Groupbox, eines Menüs, Seitenreiters oder Titel des Rahmens). Diese Meldung erfolgt nur, wenn sich der aktuelle Containername oder das Label vom dem des zuletzt angesagten

- Objektes unterscheidet.
2. Das Label des fokussierten Objekts, sofern eines existiert.
 3. Der lokale Rollenname des Objekts, wenn sich die Rolle von der zuletzt angesagten Rolle unterscheidet.
 4. Die Informationen für eine aktive Auswähländerung (Siehe [2.4, "Auswähländerung"](#)) oder eine Cursorbewegung zu einer neuen Zeile (Siehe [2.6, "Cursorbewegungen"](#)).

 **Hinweis:**

Eine Rolle ist einer der Objekttypen, die von der Accessibility-Schnittstelle unterstützt werden. Bei AT-SPI wird die Rolle als Zeichenkette ausgegeben. Diese Rollennamen sind als Konstanten in [Accessibility::Role](#) gelistet.

Beispiel 3. Fokus auf einem Button

Der `gedit`-Einstellungsdialog ist das aktive Fenster mit dem Fokus auf dem Seitenreiter „Ansicht“. Der Anwender drückt nun (**Alt+h**). Der Fokus wechselt zum „Hilfe“-Button.

Druckknopf

Direkt darauf folgt die Ausgabe für die aktive Auswähländerung.

Beispiel 4. Fokus auf einem Spinbutton

Das aktive Fenster ist der `gedit`-Einstellungsdialog mit dem Fokus auf dem Seitenreiter „Ansicht“. Der Anwender drückt nun (**Alt+e**). Der Fokus wechselt zum Spin-Knopf „Rechter Rand in Spalte“ in der Gruppe „Rechter Rand“.

Rechter Rand Rechter Rand in Spalte: Spin-Knopf

Direkt darauf folgt die Ausgabe für die aktive Auswähländerung.

Beispiel 5. Fokus auf einer Tabelle

Der `gedit`-Einstellungsdialog ist das aktive Fenster mit dem Fokus auf dem Kontrollkästchen unter dem Seitenreiter „Schrift und Farben“. Der Anwender drückt die **Tabulator-Taste**. Der Fokus wechselt zur Tabelle „Farbschema“.

Tabelle

Direkt darauf folgt die Ausgabe für die aktive Auswähländerung.

2.4 Auswähländerungen

Ein Auswähländerungsereignis (*selection change event*) kennzeichnet, dass ein Objekt nun aktiviert, selektiert oder deselektiert wurde. In den meisten Fällen wird ein Objekt aktiviert. Das aktivierte Objekt ist gewöhnlich von einem gepunkteten Rahmen umgeben. Bei einigen Kontrollelementen können auch mehrere Objekte selektiert werden. Ausgewählte Objekte werden meist farblich gekennzeichnet.

2.4.1 Aktivierung eines Objektes

Auswahlereignisse treten bei einfachen und komplexen Bedienelementen auf. Die einfachen Bedienelemente haben nicht mehr als ein sichtbares Element (z.B. Druckknöpfe). Komplexe Bedienelemente besitzen mehrere Elemente (z.B. Listen, Tabellen, Bäume).

Bei einfachen Bedienelementen folgt eine aktivierte Auswähländerung immer auf einen Fokuswechsel. Dabei wird der Text auf oder in dem Bedienelement (z.B. Druckknopf-Label) ausgewählt. Diese Auswahl kann durch den Anwender nicht beeinflusst werden.

Bei einem komplexen Bedienelement kann die Selektion auf zwei verschiedenen Wegen erfolgen: einerseits bei einem Fokuswechsel auf ein Bedienelement, in dem bereits eines seiner Objekte aktiviert ist, andererseits bei der Navigation durch die Objekte innerhalb des Bedienelementes (z.B. mit den Pfeiltasten oder mit der Maus).

Folgende Informationen werden vom `BasicSpeechScript` bei einer Auswähländerung ausgegeben:

1. Der lokale Rollenname des selektierten Objektes, wenn er sich von der zuletzt ausgegebenen Rolle unterscheidet.
2. Der Text des aktivierten Objektes, wenn er sich vom Label unterscheidet. Bei einfachen Bedienelementen entspricht der Text dem Namen des Bedienelementes. Bei komplexen Bedienelementen entspricht er dem Namen des aktiven Unterobjektes.
3. Der Kopf der aktuellen Tabelle oder des Listenelements. Ein Zeilen- oder Spaltenkopf wird nicht angesagt, wenn er nicht existiert oder mit dem vorher ausgegebenen Objekt übereinstimmt. Die Meldung erfolgt in einem der folgenden drei Formate, abhängig von den verfügbaren Informationen:
 - Zeile *% (Zeilenkopf)*, Spalte *% (Spaltenkopf)*
 - Zeile *% (Zeilenkopf)*
 - Spalte *% (Spaltenkopf)*
4. Der Status des Objektes, wenn das Objekt eines der folgenden Status-Paare besitzt:
 - aktiviert / deaktiviert
 - expandiert / zugeklappt
 - ausgewählt / nicht ausgewählt
 - animiert / nicht animiert
5. Die Baumebene des Objektes, wenn sich das Objekt in einem Baum befindet und wenn sich die Ebene von der zuletzt ausgegebenen Ebene unterscheidet. Die Wurzel-Ebene beginnt dabei mit der 1 und wird im folgendem Format ausgegeben:
 - Level *% (Ebene)*
6. Der eindimensionaler Listenindex oder der zweidimensionaler Tabellenindex des Objektes, wenn das Objekt zu einem komplexen Bedienelement gehört und der Index nicht der gleiche ist wie der zuletzt ausgegebene Index. Der Index beginnt zahlenmäßig mit der 1. Die Meldung erfolgt in einem der beiden folgenden Formaten, abhängig davon, ob es sich um eine Liste oder eine Tabelle handelt:
 - Objekt *% (Index)*
 - Objekt *% (Zeilenindex)*, *% (Spaltenindex)*
7. Der maximale Listen- oder Tabellenindex, wenn es sich um die erste Auswähländerung in einem komplexen Bedienelement handelt. Die Meldung erfolgt in einem der beiden folgenden Formate, abhängig davon, ob es sich um eine Liste oder eine Tabelle handelt:
 - von *% (Index)*
 - von *% (Zeilenindex)*, *% (Spaltenindex)*
8. Das Tastenkürzel zum Auslösen des Objektes, sofern eines existiert.
9. Der numerische Wert eines Elements (z.B. bei einem Schieberegler), wenn er nicht zuvor als Objektinhalt ausgegeben wurde.
10. Minimum, Maximum und Schrittgröße eines numerischen Bedienelementes, sofern das Element einen Wert hat. Die Meldung erfolgt in einem der beiden folgenden Formate, abhängig von den vorhandenen Informationen:
 - *% (Minimum)* bis *% (Maximum)* Schritt *% (Schrittgröße)*
 - *% (Minimum)* bis *% (Maximum)*

Beispiel 6. Aktivierte Selektion auf einem Druckknopf

Der Button erhält den Fokus wie im [Beispiel 3, "Fokus auf einem Button"](#). Gleich danach wird der Button-Name selektiert und folgende Meldung ausgegeben:

```
Hilfe Alt+H
```

Beispiel 7. Aktivierte Selektion auf einen Spin-Button

Der Spin-Knopf erhält den Fokus wie im [Beispiel 4, "Fokus auf einem Spin-Button"](#). Gleich danach wird die Textzeile des Spin-Buttons selektiert und es wird folgende Meldung ausgegeben:

```
80 1.0 bis 160.0
```

Beispiel 8. Aktivierte Selektion auf eine Tabellenzelle

Die Tabelle erhält den Fokus wie im [Beispiel 5, "Fokus auf einer Tabelle"](#). Anschließend wird die erste Tabellenzelle ausgewählt.

```
Tabellenzelle Klassisch Klassisches Farbschema Objekt 1 von 4
```

2.4.2 Objekte aus- und abwählen

Auswahl- und Abwahlereignisse (selection change) treten in komplexen Bedienelementen auf, wenn mehrere Objekte ausgewählt werden können. Wird ein zusätzliches Objekt aus- oder abgewählt, meldet das `BasicSpeechScript`:

1. Das ausgewählte Objekt in einem der folgenden Formate:
 - ausgewählt `%(Objekttext)`
 - nicht ausgewählt `%(Objekttext)`
2. Die Anzahl der aktuell selektierten Objekte in folgendem Format:
`%(Anzahl) ausgewählt`

Hinweis:

Mehrfachselektionen werden nicht angesagt.

Beispiel 9. Selektierte Objektliste

Der Datei-Browser ist aktiv. Der Fokus befindet sich in der Dateiliste und dessen erstes Objekt „doc“ ist aktiviert. Momentan sind keine Objekte selektiert. Wenn der Anwender nun **(Strg+Leertaste)** drückt, um das Objekt zu selektieren, kommt folgende Ausgabe:

```
selektiert doc 1 ausgewählt
```

2.4.3 Text aus- und abwählen

Textauswahlereignisse entstehen in Textfeldern, deren Inhalt bearbeitet werden kann. Wenn Text aus- oder abgewählt wird, liest das `BasicSpeechScript` folgende Informationen vor:

1. Die Summe der Zeichen, die markiert sind, in einen der beiden folgenden Formate:
 - ausgewählt `%(Anzahl)`
 - nicht ausgewählt `%(Anzahl)`

Der eigentliche aus- oder abgewählte Text wird vor dieser Meldung infolge der Bewegung des Textcursors ausgegeben (Siehe [2.6, "Cursorbewegungen"](#)).

2.5 Zustands- und Wertänderungen

Ein Zustands- oder Wertänderungsereignis signalisiert einen neuen booleschen oder numerischen

Wert in einem aktiven Objekt. Tritt eines dieser Ereignisse auf, verkündet das

`BasicSpeechScript`:

1. Den Zustand des Objektes, wenn das Objekt eines der folgenden Zustandspaare besitzt:
 - aktiviert / deaktiviert
 - expandiert / zugeklappt
 - ausgewählt / nicht ausgewählt
 - animiert / nicht animiert
2. Den neuen numerischen Wert des Objektes, wenn sich der Wert geändert hat.

2.6 Cursorbewegungen

Ereignisse durch Cursorbewegungen treten immer dann auf, wenn Text in ein fokussiertes, selektiertes oder editierbares Bedienelement eingefügt wird und sich dadurch die Cursorposition verändert.

Bewegt sich der Cursor dabei innerhalb einer Zeile bewegt, liest das `BasicSpeechScript`:

1. Den Text, der durch die Cursorbewegung durchlaufen wird.
2. Informationen zur Textauswahl (Siehe [2.4.3, "Text aus- und abwählen"](#)).

Wenn sich der Cursor über mehrere Zeilen hinweg bewegt oder das erste Mal in einer neuen Zeile erscheint, liest das `BasicSpeechScript`:

1. Den Text der ganzen Zeile, in der sich der Cursor befindet.
2. Informationen zur Textauswahl (Siehe [2.4.3, "Text aus- und abwählen"](#)).

2.7 Text bearbeiten

Textbearbeitungsereignisse treten auf, wenn Text in ein fokussiertes Texteingabefeld eingegeben oder dort gelöscht wird. Bei einem kurzen Textbereich, ungefähr ein Absatz oder weniger, verkündet das `BasicSpeechScript`:

1. *Rückwärts, Löschen, Ausschneiden, Kopieren* oder *Einfügen*, abhängig von der jeweiligen Aktion, die diese Textänderung ausgelöst hat. Die Aktion wird nicht angesagt, wenn Text programmatisch oder durch direktes Eintippen eingegeben wird.
2. Der eingegebene oder gelöschte Text.

Hinweis:

Ausschneiden, Kopieren und *Einfügen* werden nicht angesagt.

Wenn ein langer Text in ein Textfeld eingegeben oder dort heraus gelöscht wird, sagt das `BasicSpeechScript`:

1. Das lokale Wort für *Rückwärts, Löschen, Ausschneiden, Kopieren* oder *Einfügen*, abhängig von der jeweiligen Aktion, die diese Textveränderung ausgelöst hat.
2. Die Zeichenlänge des eingegebenen oder gelöschten Textes.

2.8 Screenreader beenden

SUE kann mit **(Alt+Shift+q)** beendet werden. Wird der Screenreader normal beendet, so kommt folgende Ausgabe vom System:

1. Die lokalisierte Verabschiedung im Format:
SUE beenden

2.9 Beispiel: Nautilus

Das folgende Beispiel zeigt, wie das `BasicSpeechScript` auf Aktionen im Nautilus-Dateibrowser antwortet. [Abbildung 1, "Nautilus - Dateibrowser-Fenster"](#) zeigt den Ausgangszustand des Dateibrowser-Fensters.

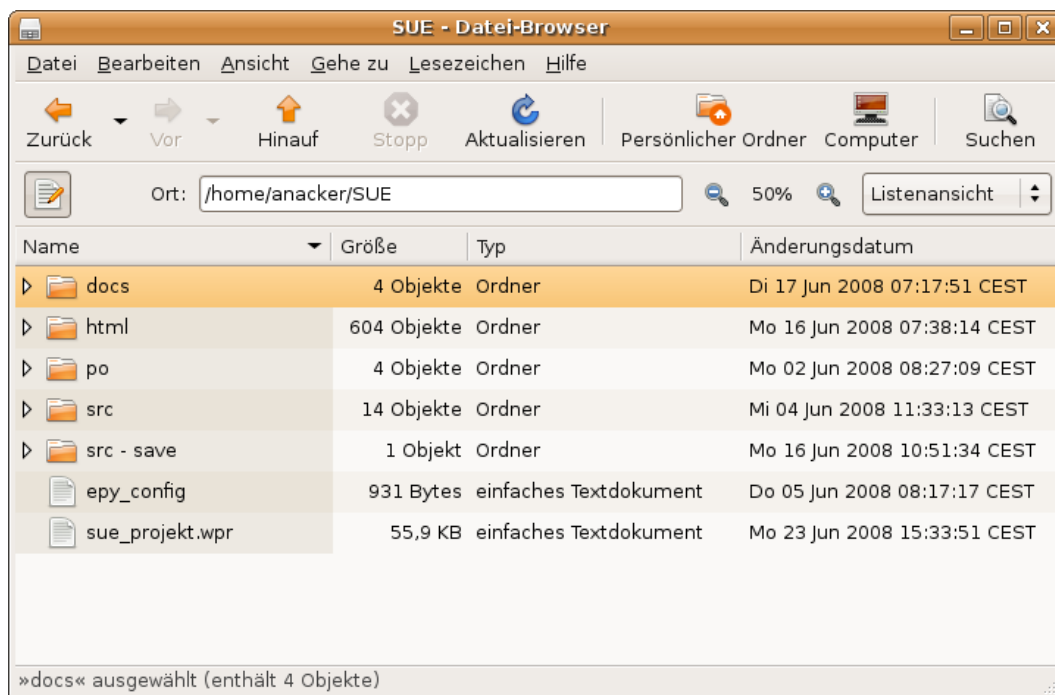


Abbildung 1: Nautilus - Dateibrowser-Fenster

Der geöffnete Ordner SUE beinhaltet 4 Unterordner und 2 Dateien. Die erste Reihe wurde ausgewählt und die erste Zelle ist aktiv.

Beispiel 10. Nautilus-Fenster wird geöffnet

Wenn das Nautilus-Fenster geöffnet wird, verkündet das `BasicSpeechScript` folgende Informationen:

Fenster	Rolle des aktivierten Oberelements.
SUE - Datei-Browser	Titel des Fensters.
Tabelle	Rolle des fokussierten Elements.
Tabellenzelle	Rolle des aktiven Objektes.
Spalte Name	Spaltenkopf des aktiven Objektes.
docs	Name des aktiven Objektes.
Objekt 1, 1 von 7, 4	Reihen- und Spaltenindex des Objektes und Umfang des Tabellenelements.
zugeklappt	Zustand des Objektes.

Beispiel 11. Nautilus-Zeilennavigation

Drückt der Anwender (**Pfeiltaste runter**), um zum nächsten Objekt zu gelangen, so gibt das `BasicSpeechScript` folgende Informationen aus:

html	Name des aktiven Objektes.
Zelle 2, 1	Reihen- und Spaltenindex des Objektes.
zugeklappt	Zustand des Objektes.

Beispiel 12. Nautilus-Ordner einklappen und ausklappen

Wird (**Shift+Pfeiltaste rechts**) gefolgt von (**Shift+Pfeiltaste links**) gedrückt, um den Ordner „html“ aus- und einzuklappen, dann sagt das `BasicSpeechScript`:

expandiert	Erste Zustandsänderung.
zugeklappt	Zweite Zustandsänderung.

Beispiel 13. Nautilus-Spaltennavigation

Wird (**Pfeiltaste links**) einmal gedrückt, dann verkündet das `BasicSpeechScript`:

Spalte Änderungsdatum	Spaltenkopf des aktiven Objektes.
Di 17 Jun 2008 07:17:51 CEST	Text des aktiven Objektes.
Zelle 1, 4	Reihen- und Spaltenindex des aktiven Objektes.

Beispiel 14. Nautilus *type-ahead*-Suche

Im letzten Beispiel beginnt der Anwender die Zeichenkette „sue_project“ einzugeben, die GTK-„type-ahead“-Suchbox erscheint und das `BasicSpeechScript` sagt:

text	Rolle des fokussierten Elements.
s u e	Text der eingegeben wird, Zeichen für Zeichen.

3 Tastenkombinationen

Wenn blinde Menschen mit Desktop-Anwendungen interagieren, dann nutzen sie die Tastatur-Navigation, um die Elemente zu bedienen. Das `BasicSpeechScript` kommuniziert diese Interaktion entsprechend des in [Abschnitt 2. "Ausgaben von Ereignissen"](#), vorgestellten Konzeptes.

Die Rückmeldung über eine erfolgte Interaktion zwischen dem Anwender und der Anwendung reicht jedoch nicht aus. Blinde und sehbehinderte Anwender brauchen auch eine Möglichkeit, den Bildschirm ohne Auswirkungen zu erkunden, so wie ein sehender Anwender die GUI mit seinen Augen untersucht. Das `ReviewScript` definiert eine Tastaturschnittstelle, die eine Übersicht über die sichtbaren Bildschirminhalte ermöglicht. Seine Tastensteuerung beinhaltet zahlreiche Kommandos, um die Programminformationen, die nicht automatisch ausgegeben werden, zu erkunden. Ein Beispiel für den Funktionsumfang ist die Ausgabe der Schrifteigenschaften an der aktuellen Textposition.

Zwei *points of regard* (PORs - Betrachtungspunkte) entscheiden, wie die Tasten ihre Funktionen

ausführen. Ein POR entspricht dem *Cursor* und der andere ist der *Pointer*.

3.1 Der Cursor

Der Cursor-POR befindet sich immer im aktuellen Element in der Anwendung, die gerade den Fokus hat. Bei Bedienelementen mit sichtbarer Textmarke bewegt sich der Cursor-POR mit der Textmarke mit. Er ruht auf dem Zeichen unterhalb eines Block-Cursors oder rechts von einer Zeilenmarke. Bei Bedienelementen, die das Auswählen von Objekten unterstützen, läuft der Cursor mit der aktiven Auswahl mit. Er ruht währenddessen auf dem ersten Zeichen des aktiven Objekts. Bei allen anderen einfachen Bedienelementen befindet sich der Cursor auf dem ersten Zeichen des einzigen enthaltenen Objektes.

Die Anwendung bestimmt, welche Elemente vom Cursor erreicht werden können. Zum Beispiel kann ein Anwender den Cursor zu nicht einem statischen Text-Label bewegen, wenn dieses vom Anwendungsentwickler nicht dafür vorgesehen wurde. Die Anwendung legt auch die verfügbaren Tasten für das Bewegen des Cursors fest. Die Tab-Taste zum Beispiel steuert den Widget-Fokus zwischen den fokussierbaren Elementen einer Anwendung.

Das `ReviewScript` macht bei diesen Regeln eine einzige Ausnahme: Die Funktion `route pointer to cursor` setzt den Cursor, sofern möglich, auf die aktuelle Pointer-Position.

Wichtig:

Wenn der Cursor seine Position ändert, wird automatisch die Pointer-Position synchronisiert.

3.2 Der Pointer

Der Pointer-POR kann zu einem beliebigen Bedienelement in einer Anwendung bewegt werden. Er ist eignet sich daher sehr gut zum Erkunden einer Bedienoberfläche ohne irgendwelche Auswirkungen. Der Anwender kann beispielsweise den Fensterinhalt lesen, indem er die Taste für das nächste Objekt drückt. Wenn der Anwender die Funktion `nächstes Objekt untersuchen` aktiviert, dann wird im `ReviewScript` der Pointer-POR auf das nächste Objekt gesetzt und das `BasicSpeechScript` gibt die entsprechenden Informationen aus. Der Anwender kann diese Funktion solange wiederholen bis das Ende der Oberfläche erreicht ist. Der Fokus der Anwendung und folglich der Cursor-POR bleiben unverändert, da diese Art der Bewegung nur den Pointer betrifft.

Eine praktische Anwendung des Pointers ist das Auslesen von Informationen, die von Seiten der Anwendung nicht als fokussierbar gelten. Zum Beispiel kann ein Programm ein statisches Text-Label mit Instruktionen beinhalten. Normalerweise können diese Elemente nicht fokussiert werden und enthaltene Hilfetexte sind damit nicht zugänglich. Hier kann jedoch der Pointer verwendet werden, um zum statischen Text zu navigieren und ihn auszulesen.

Wichtig:

Wenn der Pointer seine Position ändert, wird **nicht** automatisch die Cursor-Position synchronisiert.

3.3 Überlegungen zur Tastatur-Schnittstelle

Die Tastaturanbindung versucht die folgenden Dinge zu leisen:

- Einhaltung der Anforderungen der "Section 508", der Mindestanforderungen für barrierefreie Informationstechniken

- Minimierung der Lernkurve für den Anwender
- Minimierung der Auswirkungen auf Script-Entwicklern
- Maximierung der Bedienbarkeit / Ergonomie
- Minimierung von Konflikten zwischen den Tasten, die in Linux-basierten Anwendungen verwendet werden und den Konventionen für Linux-Tasten
- Maximierung der flexiblen Verwendung von Tastenfunktionen
- Maximierung der Portabilität der Oberfläche auf den internationalen Markt

3.3.1 Empfohlene Tastatur-Ergonomie

Die folgenden Ergonomie-Kriterien werden bei der Auswahl der Tastaturkürzel berücksichtigt. Die wichtigsten ergonomischen Bedingungen werden als erstes gelistet, nicht-ergonomische Bedingungen werden als letztes gelistet. Bedingungen, die über die Top 5 hinaus gehen, sollten ernsthaft bewertet werden, bevor sie in einem Design berücksichtigt werden.

Hinweis:

Auf einigen Laptops gibt es nur eine von verschiedenen Modifikator-Tasten. Somit ist es fast unmöglich, die Nutzung dieser Tastaturen zu optimieren, außer der Anwender weiß genau, wie er die verschiedenen Modifikator-Tasten erreichen kann.

1. Eine Kombination aus zwei Tasten, wobei beide Hände in der Ausgangsposition bleiben und die Tasten mit der entgegengesetzten Hand gedrückt werden z.B. **(Strg+Buchstabe)**.
2. Eine Kombination aus drei Tasten, wobei der Modifikator aus zwei Tasten besteht, die leicht mit einer Hand gedrückt werden können. Die andere Taste sollte mit der entgegengesetzten Hand leicht aus der Ausgangsposition erreichbar sein. Bevorzugte Kombinationen geordnet nach ihrer Präferenz: **(Alt+Strg+Buchstabe)**, **(Alt+Umschalt+Buchstabe)**, **(Alt+Caps-Lock+Buchstabe)**, **(Umschalt+Caps-Lock+Buchstabe)**, **(Strg+Caps-Lock+Buchstabe)** und **(NumPadInsert+Strg+Buchstabe)**. Alternativ sollte in dieser Kategorie die Minimierung der Handbewegung weg von der Ausgangsstellung berücksichtigt werden, um die Modifikator-Tasten zu erreichen. Auch einige nebeneinander stehende Tasten benötigen mehr Bewegung weg von der Ausgangsposition als Tastenkombinationen, die nicht nebeneinander stehen. Das ist natürlich auch abhängig vom Tastaturlayout (z.B. erfordert **(Umschalt+Caps-Lock)** auf einem ThinkPad mehr Bewegung weg von der Ausgangsposition als **(Alt+Caps-Lock)**).
3. Eine Kombination aus zwei Tasten, wo eine Hand im Bereich der Ausgangsstellung bleibt und die andere sich in einen Bereich außerhalb der Ausgangsstellung bewegt z.B. **(Caps-Lock+F12)**.
4. Eine Kombination aus drei Tasten, wobei zwei Umschalttasten leicht mit einer Hand gedrückt werden können. Die dritte Taste befindet sich außerhalb der Ausgangsposition der jeweiligen Hand. z.B. **(Alt+Strg+F12)**.
5. Eine Kombination aus vier Tasten, wobei drei nebeneinander stehende Umschalttasten mit einer Hand gedrückt werden. Die vierte Taste sollte auf der Seite der anderen Hand liegen z.B. **(Strg+Alt+Umschalt+K)**.
6. Eine Kombination aus zwei Tasten, wobei sich eine Hand außerhalb der Ausgangsposition befindet und die andere Hand auf die gegenüberliegende Tastaturseite wandert z.B.

(Einfügen-F11).

7. Eine Kombination aus drei Tasten, wobei der Modifikator aus zwei Tasten besteht, die leicht mit einer Hand gedrückt werden können. Die dritte Taste liegt auf der gegenüberliegenden Seite der anderen Hand außerhalb der Ausgangsposition z.B. **(Alt+Caps-Lock+F1)**.
8. Eine Kombination aus vier Tasten, bei der der Modifikator aus drei Tasten besteht und schwierig mit einer Hand zu erreichen ist. Die dritte Taste liegt auf der gegenüberliegenden Seite der anderen Hand außerhalb der Ausgangsposition. z.B. **(Strg+Umschalt+Caps-Lock+F1)**.
9. Eine Kombination aus drei Tasten, wobei der Modifikator aus zwei Tasten besteht, die schwer mit einer Hand zu erreichen sind. z.B. **(Strg+Einfügen)**.
10. Eine Kombination aus vier Tasten, wobei drei Umschalt-Tasten schwer mit einer Hand gedrückt werden können.
11. Eine Kombination aus fünf oder mehr Tasten.

3.4 Tastaturanbindung

Die Tastatursequenzen von SUE rufen bestimmte Funktionen im `ReviewScript`, `BasicSpeechScript`, `DeveloperScript`, `DefaultDialogScript` und weiteren Scripten im SUE-Ordner auf. Jede Tastenkombination beginnt mit einer oder mehreren Tasten als Modifikator, gefolgt von einer weiteren Taste. Eine Funktion, die an eine Tastenkombination gebunden ist, wird ausgelöst, sobald die erste der gedrückten Tasten *losgelassen* wird. Diese Vorgehensweise ist beabsichtigt, da sie Probleme durch ungewolltes wiederholtes Tastendrücken (z.B. wiederholtes Auslösen gedrückter Tasten bei Anwendern mit schwacher Handmotorik) verhindert. Für die zu drückenden Tasten werden intern die *Scan Codes* verwendet, eindeutige Bezeichner der Tasten auf der Tastatur, nicht die Tastensymbole, also deren Beschriftung. Dadurch befinden sich die Tasten immer an der gleichen Stelle auf der Tastatur, unabhängig davon, welches Layout eingestellt und welcher Buchstabe tatsächlich mit der Taste assoziiert ist.

Die grundlegenden Screenreader-Funktionen sind an **(Alt+Shift)**-Kombinationen gebunden. Dies verhindert Konflikte mit bestehenden Tastenkombinationen in Standard-Anwendungen und Features für die Tastaturzugänglichkeit. So soll beispielsweise die Verwendung von Standard-Modifikatoren der unbeabsichtigten Nutzung des *Sticky Keys*-Features vorbeugen (Siehe [Wikipedia: Sticky Keys](#)).

Die Funktionen im `DeveloperScript` sind an **(Alt+Caps-Lock)**-Kombinationen gebunden. Die Verwendung eines separaten Modifikators unterscheidet diese Tastenkombinationen von den regulären Tastenkombinationen, da das `DeveloperScript` ein Hilfsmittel für Entwickler und kein SUE-Skript im engeren Sinne darstellt.

Skript-Schreibern empfehlen wir, die **Caps-Lock**-Taste als Modifikator zu verwenden. Dies hilft Anwendern zwischen Standard- und anwendungsspezifischen Funktionen zu unterscheiden.

Hinweis:

- Einige Anbindungen folgen Ausnahmeregelungen:
1. Die Funktion `stop now` ist nur an **(Strg)** gebunden.

2. Die Funktionen im `BookmarkScript` nutzen (**Caps-Lock**) in ihrem Modifikator.
3. Die Developer-Funktion `toggle mute` ist an (**Strg-link+String-rechts**) gebunden.

Alle aktuell verfügbaren Tastenkombinationen können mit (**Alt+Shift+F4**) über den Tastenkürzeldialog abgerufen werden. Die Liste beinhaltet sowohl die Tastenkombinationen, die von den SUE-Basisskripten bereitgestellt werden als auch diejenigen Kürzel, die in anwendungsspezifischen Skripten definiert sind. Siehe auch [4.2. "Hilfsdialog"](#).

3.5 ReviewScript-Funktionen

Die folgende Tabelle listet alle Funktionen mit ihren jeweiligen Tastenkürzeln auf, die vom `ReviewScript` bereitgestellt werden. Sofern nicht anders angegeben, arbeiten alle Funktionen mit dem Pointer-POR.

Tabelle 1. Review-Funktionen und ihre Tastenkürzel

Funktion	Tastenkürzel	Beschreibung
<code>review previous item</code>	Alt+Shift+u	Bewegt den Pointer zum Anfang des vorhergehenden Objekts. Stoppt, wenn das erste Objekt der aktuellen Anwendung erreicht ist.
<code>review current item</code>	Alt+Shift+i	Bewegt den Pointer zum Anfang des aktuellen Objektes.
<code>review next item</code>	Alt+Shift+o	Bewegt den Pointer zum Anfang des nächsten Objektes. Stoppt, wenn das letzte Objekt der aktuellen Anwendung erreicht ist.
<code>review previous word</code>	Alt+Shift+j	Bewegt den Pointer zum Anfang des vorhergehenden Wortes. Stoppt, wenn das erste Wort der aktuellen Anwendung oder des aktuellen Elements erreicht ist, abhängig von der jeweiligen Einstellung.
<code>review current word</code>	Alt+Shift+k	Bewegt den Pointer zum Anfang des aktuellen Wortes.
<code>review next word</code>	Alt+Shift+l	Bewegt den Pointer zum Anfang des nächsten Wortes. Stoppt, wenn das letzte Wort der aktuellen Anwendung oder des aktuellen Elements erreicht ist, abhängig von der jeweiligen Einstellung.
<code>review previous char</code>	Alt+Shift+m	Bewegt den Pointer zum vorhergehenden Zeichen. Stoppt beim ersten Zeichen der aktuellen Anwendung oder des aktuellen Elementes, abhängig von der jeweiligen Einstellung.
<code>review current char</code>	Alt+Shift+,	Pointer bleibt beim aktuellen Zeichen.
<code>review next char</code>	Alt+Shift+.	Bewegt den Pointer zum nächsten Zeichen. Stoppt beim letzten Zeichen der aktuellen Anwendung oder des aktuellen Elements, abhängig von der jeweiligen Einstellung.
<code>focus to por</code>	Alt+Shift+p	Versucht standardmäßig den Cursor zum Pointer-POR zu bewegen. Sofern möglich, werden der Fokus der Anwendung, die aktive Auswahl und der Cursor auf die

Funktion	Tastenkürzel	Beschreibung
		Position des Pointers gesetzt. Wenn dies nicht möglich ist, wird eine Meldung ausgegeben, dass der Cursor nicht bewegt werden kann.
pointer to por	Alt+Shift+ö	Versucht standardmäßig den Pointer auf den Cursor-POR zu setzen.
mouse to por	Alt+Shift+-	Versucht den Maus-Zeiger auf die Position des Pointer-PORs zu setzen.

3.6 BasicSpeechScript-Funktionen

Die folgende Tabelle listet die Funktionen mit ihren jeweiligen Tastenkürzeln auf, die vom BasicSpeechScript bereitgestellt werden. Einige dieser Funktionen sind übrigens mit Funktionen im ReviewScript verknüpft und werden durch die Tastenkürzel im ReviewScript ausgelöst.

Sofern nicht anders angegeben, beziehen sich alle Funktionen auf den Pointer-POR.

Tabelle 2. Grundlegende Sprach-Funktionen und ihre Anbindungen

Funktion	Anbindung	Beschreibung
stop now	Strg	Stoppt die aktuelle Sprachausgabe.
increase speech rate	Alt+Shift+Bild hoch	Erhöht die Sprechgeschwindigkeit. Wird diese Funktion nicht vom Ausgabegerät unterstützt, passiert nichts.
decrease speech rate	Alt+Shift+Bild runter	Verringert die Sprechgeschwindigkeit. Wird diese Funktion nicht vom Ausgabegerät unterstützt, passiert nichts.
read review item, read review skip	review previous item, review current item, review next item	Liest das Objekt an der aktuellen Pointer-Position vor. Ist das Objekt leer, wird es nur dann vorgelesen, wenn die Überspringen-Einstellung im ReviewScript auf „Report“ gesetzt ist.
read review word, spell review word, pronounce review word	review previous word, review current word, review next word	Liest das Wort an der aktuellen Pointer-Position. Die Funktion wechselt zwischen Vorlesen, Buchstabieren und Phonetisieren (Siehe Wikipedia: Buchstabieralphabet) des aktuellen Wortes, wenn die Funktion review current word mehrmals ohne Unterbrechung aufgerufen wird.
read review char, spell review char, pronounce review char	review previous char, review current char, review next char	Liest das Zeichen an der aktuellen Pointer-Position vor. Die Funktion wechselt zwischen Vorlesen und Phonetisieren (Siehe Wikipedia: Buchstabieralphabet) des aktuellen Zeichens, wenn die Funktion review current char mehrmals ohne Unterbrechung aufgerufen wird.
read top	Alt+Shift+t	Setzt den Pointer auf das erste Element der aktuellen Ansicht und gibt es aus. Normalerweise ist dies der Titel des aktiven Fensters.

Funktion	Anbindung	Beschreibung
<code>read bottom</code>	Alt+Shift+n	Setzt den Pointer auf das letzte Element der aktuellen Ansicht und gibt es aus. Normalerweise ist dies die Statuszeile des aktiven Fensters.
<code>read description</code>	Alt+Shift+d	Liest die Beschreibung des aktuellen Elements.
<code>read text color, read text attributes</code>	Alt+Shift+f	Liest verschiedene Textattribute vor. Diese Funktion wechselt zwischen Farb- und Textattributen. Zuerst werden Schrift- und Hintergrundfarbe ausgegeben, gefolgt von den jeweiligen RGB-Werten (Rot, Grün, Blau). Beim nächsten Drücken des Tastenkürzels werden alle übrigen Textattribute im Format als Name-Wert-Paare ausgegeben.
<code>read all</code>	Alt+Shift+Pfeiltaste runter	Liest ab der aktuellen Position alle vorhandenen Informationen vor.
<code>where am i now, where am i ancestors</code>	Alt+Shift+?	Gibt die aktuelle Position aus. Diese Funktion wechselt zwischen zwei verschiedenen Ausgaben. Beim ersten Drücken werden detaillierte Informationen über das aktuelle Element ausgegeben. Beim zweiten Drücken werden verkürzte Detailinformationen über jeden Vorgänger bis zum Wurzelement der aktiven Anwendung ausgegeben.

Die Navigation über Elemente folgt dem gleichen Ausgabemuster wie die Auswahlereignisse. Siehe [2.4.1, "Aktivierung von Objekten"](#).

3.7 BookmarkScript-Funktionen

In der folgenden Tabelle werden die Funktionen mit ihren Tastenkürzeln aufgelistet, die vom `BookmarkScript` bereit gestellt werden.

Tabelle 3. Lesezeichen-Funktionen und ihre Tastenkürzel

Funktion	Tastenkürzel	Beschreibung
<code>bookmark reg add gesture</code>	Alt+Caps-Lock+1 bis Alt+Caps-Lock+0	Speichert den aktuellen Pointer-POR im angegebenen Slot (1-0). Wenn der Slot bereits ein Lesezeichen enthält, wird nach einer Bestätigung gefragt. Das erneute Drücken des Tastenkürzels bestätigt das Überschreiben.
<code>bookmark reg goto gesture</code>	Caps-Lock+1 bis Caps-Lock+0	Setzt den Pointer auf das gespeicherte Lesezeichen im angegebenen Slot (1-0).
<code>bookmark reg where am i gesture</code>	Alt+Shift+1 bis Alt+Shift+0	Vergleicht das gespeicherte Lesezeichen im angegebenen Slot (1-0) mit der Wurzel der Ansicht.
<code>bookmark reg bm compare gesture</code>	Alt+Caps-Lock+Shift+1 bis Alt+Caps-Lock+Shift+0	Vergleicht das gespeicherte Lesezeichen im angegebenen (1 – 0) Slot mit dem aktuellen Pointer.

3.8 SearchScript-Funktionen

Die folgende Tabelle listet die Funktionen und die jeweiligen Tastenkürzel auf, die vom SearchScript bereitgestellt werden.

Tabelle 4. Such-Funktionen und ihre Tastenkürzel

Funktion	Tastenkürzel	Beschreibung
search show chooser	Alt+Shift+s	Öffnet bzw. schließt den Suchdialog. (Siehe 4.3, "Suchdialog")
search find next	Alt+Shift+e	Sucht das nächste Element.
search find previous	Alt+Shift+w	Sucht das vorherige Element.

3.9 DefaultDialogScript-Funktionen

Die folgende Tabelle listet die Funktionen mit ihren jeweiligen Tastenkürzeln auf, die vom DefaultDialogScript bereitgestellt werden.

Tabelle 5. Dialog-Funktionen und Tastenkürzel

Funktion	Tastenkürzel	Beschreibung
show settings chooser	Alt+Shift+F2	Öffnet den SUE-Einstellungsdialog. (Siehe 4.1, "SUE-Einstellungsdialog")
show script chooser	Alt+Shift+F3	Öffnet den Script-Einstellungsdialog (Siehe 4.4, "Script-Einstellungsdialog")
show command chooser	Alt+Shift+F4	Öffnet eine Übersicht über alle derzeit verfügbaren Tastenkürzel (Siehe Tastenkürzeldialog)

3.10 DeveloperScript-Funktionen

Die folgende Tabelle listet die Funktionen und die jeweiligen Tastenkürzel auf, die vom DeveloperScript bereitgestellt werden.

Tabelle 6. Developer-Funktionen und Tastenkürzel

Funktion	Tastenkürzel	Beschreibung
toggle mute	Strg links+Strg rechts	Schaltet die Sprachausgabe aus bzw. ein.
say scripts	Alt+Caps-Lock+j	Gibt alle verwendeten Scripte für eine Anwendung aus. Beginnt mit dem Script, das ein Ereignis als erstes abarbeitet.
reload scripts	Alt+Caps-Lock+k	Lädt alle Scripte für eine Anwendung neu. Eine Veränderung am Scriptcode kann nach dem Neuladen sofort ausgeführt werden.
toggle monitors	Alt+Caps-Lock+l	Schaltet die Monitore, die für ein Profil registriert sind, ein bzw. aus. (Siehe 4.5, "Entwicklungsdialoge")

3.11 Anwendungsspezifische Tastenkürzel

Anwendungsspezifische Skripts können zusätzlich eigene Tastenkürzel definieren. Dieses Dokument behandelt jedoch nur diejenigen Kommandos, die in den Basisskripten definiert sind. Zusätzliche Tastenkürzel werden im [SUE-Wiki](#) und im [SUE-Nutzerhandbuch](#) beschrieben.

// Der verbleibende Teil des Dokumentes wurde noch nicht überarbeitet.

4 Grafische Dialoge

Not all interactions with the user can effectively occur via automatic event reporting and keyboard commands. For instance, having keys to configure all Script and device settings is untenable, especially since each Script can define new user options. Instead, the screen reader interface has GUI dialogs to enable more complex interactions with users.

4.1 Settings Chooser

The settings chooser allows the user to configure options for Scripts, devices, the LSR platform, and the current user profile. The chooser has four tab panels that display the settings in each of these areas. The widgets of the first three panels are generated dynamically based on the options exposed by Scripts, devices, and the LSR platform. The structure of the fourth panel is fixed at design time to show the contents of the active user profile.

All changes to settings take affect immediately in the user interface. Setting values are made permanent by activation of the Apply or OK buttons. Setting values are discarded on activation of the Cancel button. The dialog closes immediately when OK is activated.

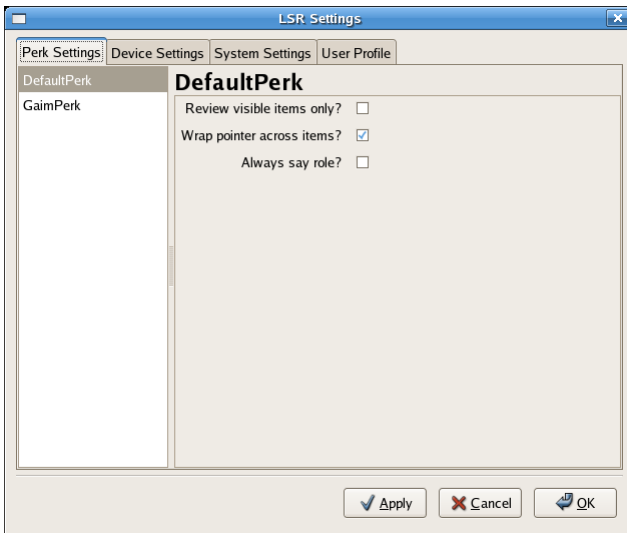
Two hotkeys are defined to enable quick navigation of the dialog. **Alt-1** gives focus to the row of tabs for navigating to the four configuration panels. **Alt-2** gives focus to the first focusable control in the current panel.

4.1.1 Script Settings

The Script Settings tab panel has two primary areas. On the left, a list box shows the localized names of Scripts that are either currently loaded or associated with the active user profile. Only Scripts that have user configurable settings are shown. Selecting one of the names in the list causes the scroll panel on the right to update. This panel displays widgets for configuring the settings for the selected Script. See [Section 5.1, "Script Settings"](#) for a list of the settings provided by the core screen reader Scripts.

The settings for a given Script name effect all instances of that Script. In other words, the settings are global.

Figure 2. Script settings panel



4.1.2 Device Settings

The Device Settings tab panel also has two primary areas. The root level of the tree on the left shows the localized names of devices that are currently loaded. Selecting one of the names in the list causes the panel on the right to update. The panel contains widgets for configuring the default settings for the selected device.

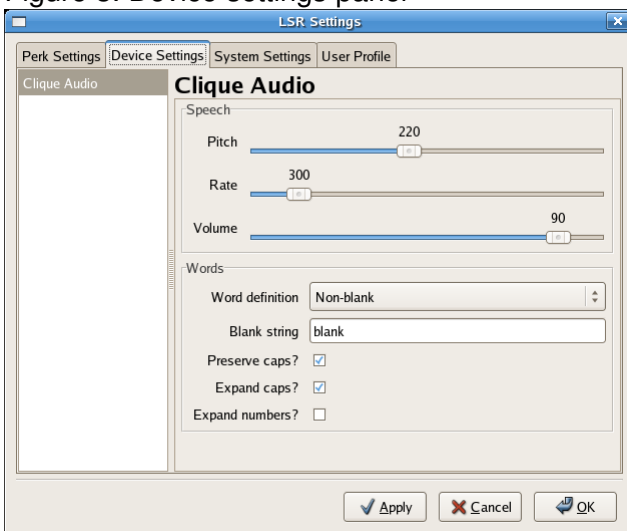
If a device supports fine-grained configuration of output, a second level of tree nodes appears under the device name. The second level items have localized names for the semantic tags supported by LSR. When selected, the panel to the right updates with widgets for configuring settings for the selected tag. For instance, say a user selects the Color node under a speech device. The panel on the right should show options for changing the voice characteristics used when speaking color information.

 Note

Semantic tags are not configurable, only default device settings.

The available settings are defined by the capabilities of the device being configured. See [Section 5.2, "Device Settings"](#) for a list of the settings provided by typical screen reader devices.

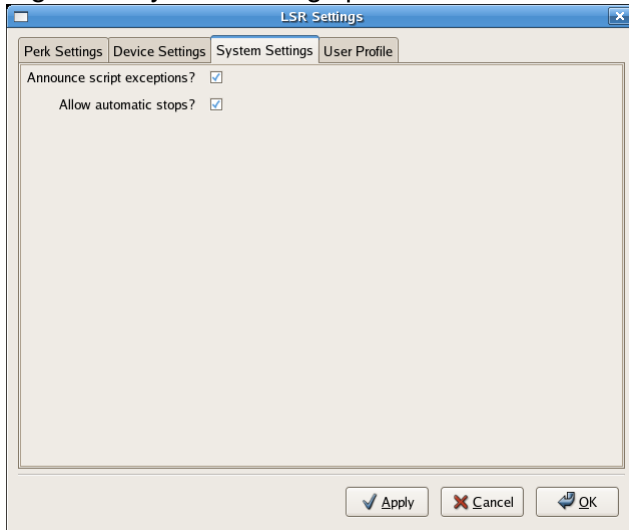
Figure 3. Device settings panel



4.1.3 System Settings

The System Settings tab panel shows widgets for options made available by the LSR platform itself, independent of any Scripts or devices. These options are typically for developers or advanced users, and effect how the LSR platform scripting API methods behave. See [Section 5.3, “System Settings”](#) for a list of the settings provided by the LSR platform.

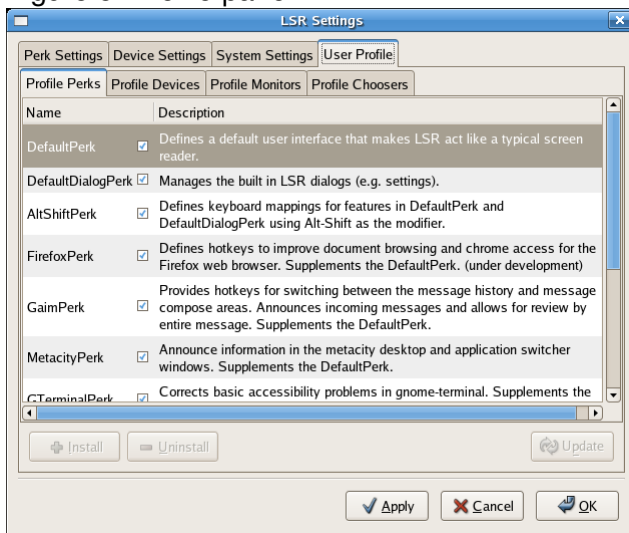
Figure 4. System settings panel



4.1.4 Profile Settings

The profile tab panel shows the available extensions which the user can set to load automatically. The tab is labeled with the name of the active profile for reference (e.g. user). The profile panel has four sub-tab panels representing the four kinds of LSR extensions: Scripts, devices, monitors, and choosers.

Figure 5. Profile panel



4.1.4.1 Installing Extensions

The user may install any kind of extension by activating the Install button. The table of extensions of the given type is automatically updated to show the newly installed item. However, the actual install does not happen unless the user activates the OK or Apply button. The new extension is available only to the current user, not to all users of the operating system. It defaults to loading automatically in the current user profile.

The user may uninstall any extension previously installed *by this user only* by activating the Uninstall button. The table of extensions of the given type is automatically updated to show the newly installed item. However, the actual uninstall does not happen unless the user activates the OK or Apply button.

The user may activate the Reset button to restore the current profile to a reasonable default state. All extensions packaged with the LSR platform will be returned to their default configuration. Extensions installed by the user will not be uninstalled.

 Tip

The install, uninstall, and reset buttons are disabled.

4.1.4.2 Associating Extensions

The Profile Scripts sub-panel shows a table of installed Scripts. The names of the Scripts are given in the first column along with their descriptions in the second column. Scripts with check marks next to their names are loaded automatically either when every application starts or when a particular application starts. Exactly when a Script should load is determined by its author.

 Note

The user can override the author's association at the command prompt. See **man lsr** for more information.

The Profile Devices sub-panel shows a table of installed devices. The first column enumerates their names and the second column their descriptions. LSR attempts to load devices with check marks next to their names at start-up, in top-to-bottom order. The user can change the attempted load order using the Raise and Lower buttons. The load order is executed immediately when either of the Apply or OK buttons is activated.

 Important

A device loads only if all of the services it provides are not already provided by one or more devices earlier in the load order. For instance, even if three speech devices are checked, only the first device to successfully initialize will be loaded.

The Profile Monitors sub-panel shows a table of installed monitors and their descriptions. Monitors with check marks next to their names are automatically loaded at start-up. Monitors are immediately loaded or unloaded when either of the Apply or OK buttons is activated.

The Profile Choosers sub-panel shows a table of installed choosers and their descriptions. The table is read-only since choosers do not load automatically, only under the direction of an accompanying Script.

4.2 Help Chooser

The help chooser gives usage and accessibility information about the widget at the pointer POR and a description of the features provided by all Scripts currently loaded for the active application. This information is separated into the four tab panels described below.

 Tip

The help chooser is not implemented.

4.2.1 Control Help

This panel displays instructions on how to use a known GUI control in a standard text box along with an example of the control. For example, if the pointer is on a check box when the help dialog is shown, something like the following text is provided:

A check box represents a setting that is either true or false. When it is "checked," it is true. When it is "unchecked," it is false. Press the **Spacebar** to switch between checked and unchecked. Press **Tab** or **Shift-**

Tab to navigate to the next or previous control outside of the check box.

If the control is of an unknown type, the panel shows text stating this fact. In this case, the panel contains no example widget.

4.2.2 Accessibility Info

This panel shows a text description of the accessibility information available for a GUI widget in a standard text box along with an example of the control. For example, if the pointer is on a radio button belonging to a group of three, something like the following text is given:

The role of this widget is radio button. Its accessible name is blank. Its description is blank. It has the state checked. It has a label "Two" and is a member of a group of three radio buttons.

If the control is inaccessible, the panel contains text stating this fact.

4.2.3 Command Help

The commands panel contains a table describing the commands provided by the Scripts loaded on the current application and their bindings to input gestures (e.g. keyboard combos, Braille device buttons). The first column of bindings is generated automatically by querying the loaded Scripts.

The second column of descriptions is generated by accessing the descriptions of the Tasks registered in the loaded Scripts and bound to input gestures.

4.2.4 Application Info

This panel contains a table with the names of the Scripts loaded for the active application and help on using the features of those Scripts in the active application. The first column is populated with the names of Scripts having help strings. The second column shows the help strings obtained by querying the loaded Scripts. Scripts not having help strings are not listed. For example, a Script for the Firefox web browser might suggest the user read the page using the review keys, and also explain how to navigate by elements like headers, form controls, links, and so on.

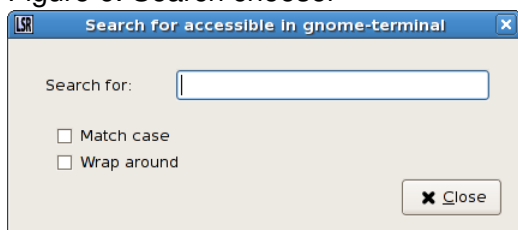
4.3 Search Chooser

The search chooser prompts the user to enter a string to locate in the active view. The dialog contains a single line text entry box, a Close button, and check boxes for matching case and wrapping search. When the dialog is shown for the first time, text entry field is empty. When shown again after a search, the text entry field contains the last string entered for this view. The text of the last search string is selected.

Tip

The last search string is not presented when the dialog is opened again.

Figure 6. Search chooser



4.4 Script Chooser

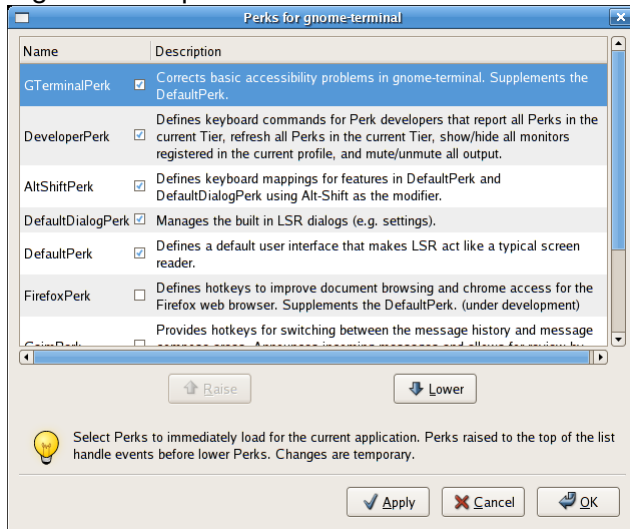
The Script chooser gives the user control over which scripts are currently loaded for the active application. The table lists the names and descriptions of all installed Scripts. Script names with a check next to them are set to load or remain loaded on the application named in the window title. Script names without checks are set to unload or remain unloaded. The actual loading and

unloading of Scripts occurs when the OK or Apply button is activated.

The order of the Scripts in the table determines the order in which loaded Scripts handle events. The Script at the top of the list handles events first while the Script at the bottom of the list handles events last. The user can change the ordering of the Scripts using the Raise and Lower buttons. Only Scripts set to load may be raised and lowered. Moving unloaded Scripts is meaningless. When a Script is checked, it is immediately raised to the top of the list. When it is unchecked, it is lowered below the last checked Script.

The hotkey **Alt-1** is defined to quickly give focus to the table of Scripts from anywhere in the dialog.

Figure 7. Script chooser window



4.5 Developer Dialogs

The developer dialogs are intended to assist LSR script writers in debugging their custom Scripts. Three monitors provide a running log of important events while the view chooser gives a developer the chance to explore and test the accessible components of an application.

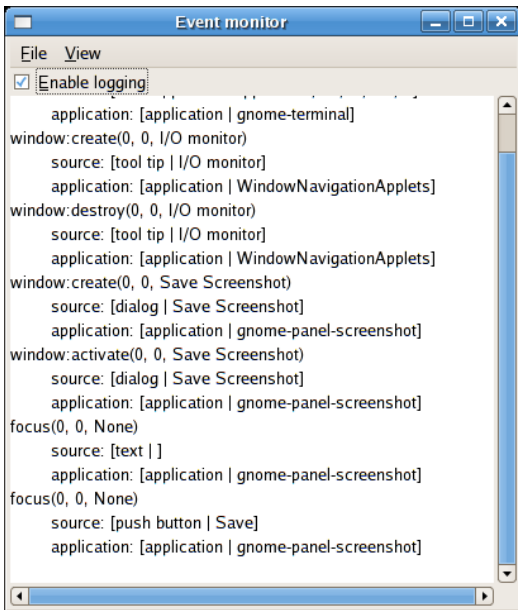
All three of the developer monitors have nearly the same layout. A large, read-only text area practically fills the window. All events are logged into this area in plain text. A check box for enabling and disabling logging resides above the text area. When unchecked, all log messages are discarded. Regardless of this setting, no events are logged when an event monitor window is active. The File menu has options for clearing and saving the current log. The View menu has checkable items for enabling or disabling the kinds of events that are logged.

4.5.1 Event Monitor

The event monitor logs the raw accessibility events occurring on the desktop. The View menu has items for all of the major event types that can be logged. The format of each entry in the log is the following:

```
%(event type)s(%(event data)s
  source: [%(accessible name)s | %(accessible role)s]
  application: [%(accessible name)s | %(accessible role)s]
```

Figure 8. Event monitor window



Note

The raw event log never includes events generated from LSR itself.

4.5.2 Task Monitor

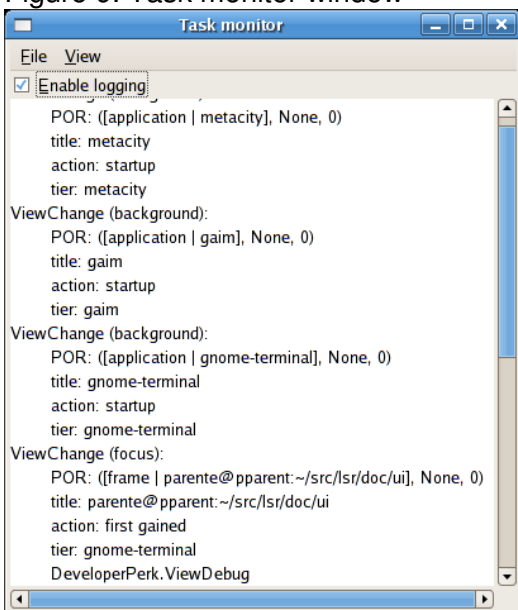
The task monitor logs the execution of Tasks registered in Scripts to handle events. The View menu has items for all of the known Task types. The format of each entry in the log is the following:

```
%(task type)s(%(task layer)s:
  %(event specific data)s
  tier: %(tier name)s
  %(Task name)s in %(Script name)s.
  propagate: %(propagate event)s
```

...

Additional reports of Tasks executing in response to events appear under the first one at the same indentation level. Tasks explicitly executed appear indented under the Task that executed them. Tasks that chain to other Tasks appear indented and in color when they are executed.

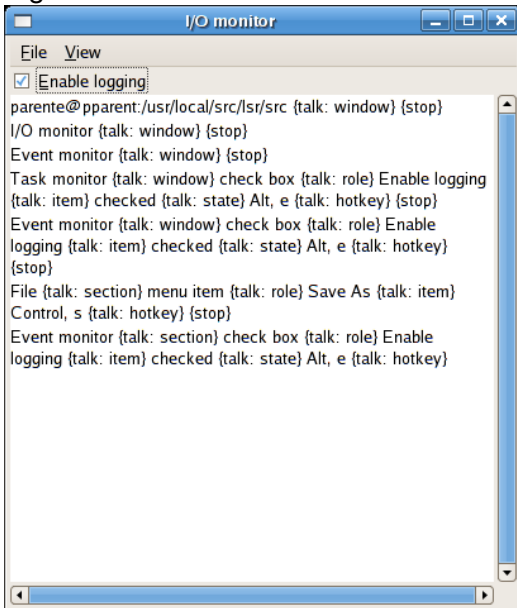
Figure 9. Task monitor window



4.5.3 I/O Monitor

The I/O monitor logs all content, semantic tags, and commands sent to an output device as well as input gestures on input devices. The View menu has items for all of the known kinds of input and output commands. The format of the I/O log differs from that of the other monitors. In keeping with the natural flow of output utterances, all information is logged on a single line. Line breaks are inserted after stops or when I/O from a different device needs to be interleaved in the log. Content sent to a device is written as plain text. Stop and talk commands, index markers, filenames, and input gestures are wrapped in braces. Talk commands also include the name of the semantic tag associated with the preceding content. The templates for each of these items are as follows:

Figure 10. IO monitor window



5 Configurable Settings

Scripts, devices, and the LSR platform itself may define settings that can be configured by a user in the settings chooser (see [Section 4.1, "Settings Chooser"](#)). The settings are defined as programmatic variables with metadata indicating its type, its human readable name, and its extended help description. The settings dialog uses this information to populate its graphical user interface with controls for manipulating the available settings.

5.1 Script Settings

The following tables list the settings defined by the ReviewScript and the BasicSpeechScript. The setting names, descriptions, type, and default values are given.

Table 7. Settings for the ReviewScript

Name	Description	Type	Default
Skip empty items	When set to a <code>always</code> , review functions silently skip over items without any text. When set to <code>Never</code> , the review functions stop on items with no text. When set to <code>Report</code> , the review functions skip over items without any text, but signal the skip so other Scripts can notify the user.	choice	report
Review invisible items	When set, review traverses items not visible on the screen. Otherwise, review walks visible items only.	boolean	false

Name	Description	Type	Default
Wrap pointer	When set, review by word or character crosses item boundaries. Otherwise, first and last are announced at item boundaries.	boolean	true

Table 8. Settings for the BasicSpeechScript

Name	Description	Type	Default
Echo words	When set, entire words are spoken when editing text.	boolean	false
Echo characters	When set, individual characters are spoken when editing text.	boolean	true
Always say role	When set, widget roles are always spoken. Otherwise, roles are announced only when a new type of widget is encountered.	boolean	false

5.2 Device Settings

The following tables list the settings common to all serial output devices, specific to audio device, and specific to Braille devices. The setting names, descriptions, type, and default values are given. A given device may support any subset of these settings depending on its capabilities. It may also allow the user to change these settings globally for the device or per semantic tag. In the latter case, a setting may be exposed relative to the global value for the device.

Table 9. Settings common to all serial output devices

Name	Description	Type	Default
Cap prefix	Text to insert before capital letters.	string	
Preserve caps	When set, capital letters are differentiated from their lowercase equivalents.	boolean	
Ignore characters	Characters to treat as blanks.	string	
Maximum repeat	The minimum number of subsequent repeated characters required before a word or item is shortened.	integer range	4
Word definition	Defines what constitutes a word for output processing. Non-blank includes all characters up to a blank. Alphanumeric and punct. includes all characters up to the first character that is not a letter, number, or punctuation mark. Alphanumeric includes all letters and numbers up to the first non-letter or non-number. Alphabetic includes all letters up to a non-letter.	string choice	Non-blank

Tip

Cap prefix and maximum repeat are not supported.

Table 10. Settings common to audio output devices

Name	Description	Type	Default
Voice	Name of a pre-defined text-to-speech synthesizer voice.	string choice	(engine dependent)
Volume	Volume as a percentage of total volume.	percent range	90%
Rate	Speech rate.	integer	(engine

Name	Description	Type	Default
		range	dependent)
Pitch	Voice fundamental pitch.	integer range	(engine dependent)
Gender	Voice gender.	string choice	(voice dependent)
Aspiration	Voice breathiness or airiness.	integer range	(voice dependent)
Frication	Voice harshness or clickiness.	integer range	(voice dependent)
Intonation	Voice tone variation.	integer range	(voice dependent)
Head size	Voice reverberation.	integer range	(voice dependent)
Channel	Audio channel number. All audio within a channel is output serially while audio across channels is output in parallel.	integer	0
Position	Three-dimensional spatial position of channel with x,z in azimuth and y in elevation.	integer tuple	(0, 0, 0)
Instrument	MIDI instrument used for playing non-speech sound.	string choice	Acoustic grand piano
Language	Language and locale of the voice to use for speech synthesis.	string choice	(engine dependent)
Expand numbers	When set, inserts spaces before numbers causing them to be read individually.	boolean	false
Expand caps	When set, inserts spaces before capital letters causing them to be read individually.	boolean	true
Blank string	String to speak to represent a blank or ignored character.	string	blank
Stoppable	When set, output can be stopped.	boolean	true
Mute speech	When set, prevents all speech output.	boolean	false
Mute non-speech	When set, prevents all non-speech sound output.	boolean	false
Mute	Prevent all audio output.	boolean	false
Continuous	Loop all audio output.	boolean	false
Say all punctuation	When set, all punctuation will be spoken. Otherwise, only punctuation deemed important by the speech engine will be pronounced.	false	
Spelling format	Describes how words should be output. Text indicates all characters in the word are output without spelling. Pronounce means all punctuation characters in the word are spelled. Spell means all characters in the word are spelled. Phonetic means all characters in the word are spelled using the NATO phonetic alphabet .	string choice	Text

5.3 System Settings

The following table list the settings specific to the LSR platform itself. The setting names, descriptions, type, and default values are given. These options are available for developers and advanced users.

Table 11. Settings for the LSR platform

Name	Description	Type	Default
Announce script exceptions	When set, important Script exceptions will be announced on output devices.	boolean	true
Allow automatic stops	When set, script events interrupt output as expected. Otherwise, script stops are ignored and all output is queued without interruption.	boolean	true

5.4 Other Settings

Custom Scripts may define settings specific to their operation. This document only covers the settings defined by the core Scripts. Additional settings are described in the Linux Screen Reader User's Guide.

Hinweis

Dieses Dokument basiert auf der "[Linux Screen Reader UI Specification](#)" von

Pete Brunet <pbrunet@us.ibm.com>

Peter Parente <pparent@us.ibm.com>

Larry Weiss <lweiss@us.ibm.com>

Copyright © 2006, 2007 IBM Corporation

Freigegeben unter der BSD-Lizenz <http://www.opensource.org/licenses/bsd-license.php>.