S@T 01.30 V1.0.5 (2001-06)

Test Specification

VALIDATION TEST PLAN

SYSTEM FUNCTIONAL TESTS

History

Document history		
Release	Approved by	Comment
1.0.0	TDG	Merge of existing input papers, prepared for M19 in Munchen
1.0.1	TDG	Adding part concerning encrypt/decrypt mechanisms and part about reference application.
1.0.1bis	TDG	Updated chapter “deck and deck level”.
1.0.2	TDG	Update after M20 in Munich
1.0.3	TDG	Update after M21 in Paris, merging with Orga’s input for M21 done.
1.0.4	TDG	Update after M22 in Munich.
1.0.5	TDG	Editorial modifications at meeting 30 for publications

1 List of documents

[1] S@T 01.10 : “S@TML, S@T markup language”

[2] S@T 01.00 : “SBC, S@T byte code”

[3] S@T 01.20 : “SSP, S@T session protocol”

[4] S@T 01.21 : “S@T administrative commands”

[5] S@T 01.22 : “S@T operational commands”

[6] GSM 03.38 : "Digital cellular telecommunications system (Phase 2+); Alphabets and language-specific information".

[7] GSM 11.11. "Digital cellular telecommunications system (Phase 2+); Specification of the Subscriber Identity Module – Mobile Equipment (SIM – ME) interface".

[8] GSM 11.14. "Digital cellular telecommunications system (Phase 2+); Specification of the SIM Application Toolkit for the Subscriber Identity Module – Mobile Equipment (SIM – ME) interface".

[9] GSM 03.48. "Digital cellular telecommunications system (Phase 2+); Security Mechanisms for the SIM Application Toolkit; Stage 2".

This document is part of a specification set, please refer to "S@T Release Note" for a comprehensive document list, including document versions.

2 Overview

This document defines which tests must be performed to ensure interoperability of browsers and gateways of the different manufacturers from the SIMalliance group. This document focuses on functional end-to-end tests, this is to say it specifies SATML pages which :

- Either are to be mediated through the whole system and have to produce a specific behaviour on the browser
- Or specifies error cases, and that an error message must appear on the browser.

In all SATML or WML pages described in this document, an header must be added when running the tests. This header must be :

- For a SATML document :
  - If the first tag in the document is “<satml>” :
    ```
    <?xml version="1.0"?>
    <!DOCTYPE satml SYSTEM "http://www.simalliance.org/DTD/satml103.dtd">
    ```
  - If the first tag of the document is “<wml>” :
    ```
    <?xml version="1.0"?>
    <!DOCTYPE wml SYSTEM "http://www.simalliance.org/DTD/satml103.dtd">
    ```
- For a WML document :
  ```
  <?xml version="1.0"?>
  <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
  "http://www.wapforum.org/DTD/wml_1.1.xml">
  ```

3 SATML Core

3.1 Decks and cards

3.1.1 Decks

3.1.1.1 TEST_DECK_LEVEL_satml.01

Description

The encoding to SMS default alphabet shall be tested. In order to make sure that SMS alphabet is used, a ME supporting only SMS alphabet may be used.

Source

```
<!-- sat-enc-type=”sms” -->

<?xml version="1.0" encoding=”ISO-8859-1”?>

<satml

     sat-storage=”static”

     sat-help=”help text with special characters £¥èé@$$”>

  <card>

     <p>

       Please request help

       <input name=”test” />

     </p>

  </card>

</satml>
```

Result

DE shall encode text string “Please request help” and ”help text with special characters £¥èé@$” in SBC using SMS default alphabet (GSM 03.48). The help text “help text with special characters £¥èé@$” must appear, if help was requested by the user. Note that the '$' sign must be escaped in the markup source. Otherwise it would be interpreted as variable reference.

3.1.1.2 TEST_DECK_LEVEL_satml.02

Description

The encoding to UCS2 ISO/IEC 10646 alphabet is tested, in case of “text string” in the STK command genrated by the browser. In order to check that UCS2 alphabet is used, a ME supporting UCS2 must be used.

Source

```
<!-- sat-enc-type=”ucs2” -->

<?xml version="1.0" encoding=”ISO-8859-1”?>

<satml

     sat-storage=”static”
```

```
    sat-help=”help text with special characters ê &#x041f;”>
  <card>
    <p>
      Please request help ê &#x041f;
      <input name=”test”/>
    </p>
  </card>
</satml>
```

Result

DE shall encode text string “Please request help ê Π” and ”help text with special characters ê Π” in SBC using UCS2 alphabet. The help text “help text with special characters ê Π” must appear, if help was requested by the user.

3.1.1.3 TEST_DECK_LEVEL_satml.03

Description

The encoding to UCS2 ISO/IEC 10646 alphabet is tested, in case of an “alpha identifier” in the STK command genrerated by the browser. In order to check that UCS2 alphabet is used, a ME supporting UCS2 must be used.

Source

```
<!-- sat-enc-type=”ucs2” -->
<satml>
  <card>
    <p>
    <sat play-tone
sat-title=”&#x041f;”/>
    </p>
  </card>
</satml>
```

Result

The text “Π” must be displayed on the ME, and tone “beep” must be played at the same time.

3.1.1.4 TEST_DECK_LEVEL_satml.04

Description

Normal context. Storage static, ie decks may be fetched from cache.

Source

```
<!-- DECK 1 -->
<satml sat-storage=”static”>
  <card>
    <p>
```

```
      deck 1
      <input title=”URL of deck 2?” name=”url” />
    </p>
    <do type=”accept”>
      <go href=”$url”/>
    </do>
  </card>
</satml>

<!-- DECK 2 -->
<satml sat-storage=”static”>
  <card>
    <p>
      deck 2
      <input title=”URL of deck 1?” name=”url” />
    </p>
    <do type=”accept”>
      <go href=”$url”/>
    </do>
  </card>
</satml>
```

Result

If deck 1 is requested within deck 2, deck 1 may be fetched from cache, if the cache mechanism is supported by the browser and no additional request to the gateway may be necessary.

3.1.1.5 TEST_DECK_LEVEL_satml.05

Description

Normal context. Storage dynamic, ie decks must not be fetched from cache.

Source

```
<!-- DECK 1 -->
<satml sat-storage=”dynamic”>
  <card>
    <p>
      deck 1
      <input title=”URL of deck 2?” name=”url” />
    </p>
    <do type=”accept”>
      <go href=”$url”/>
    </do>
  </card>
</satml>


<!-- DECK 2 -->
<satml sat-storage=”dynamic”>
  <card>
    <p>
      deck 2
      <input title=”URL of deck 1?” name=”url” />
    </p>
    <do type=”accept”>
      <go href=”$url”/>
    </do>
  </card>
</satml>
```

Result

If deck 1 is requested within deck 2, deck 1 must not be fetched from cache even if the cache mechanism is supported by the browser. Instead an additional request to the gateway is necessary.

3.1.2 Cards

3.1.2.1 TEST_DECK_LEVEL_card.01

Description

Normal context. Task is to test the end of card behaviour.

Source

```
<satml>
  <card>
    <p sat-auto-clr=”true”>Hello World</p>
  </card>
</satml>
```

Result

The text :“hello world” is displayed, and the browser switches to an implicit pause state.

3.1.2.2 TEST_DECK_LEVEL_card.02

Description

Task is to test the back behaviour, if history stack is empty

Source

```
<satml>
  <card>
    <p sat-auto-clr=”false”>Please press back button</p>
  </card>
</satml>
```

Result

Browser must show predictable and not erroneous behaviour.

3.1.2.3 TEST_DECK_LEVEL_card.03

Description

Task is to test the back behaviour, if a card is not added to history stack

Source

```
<satml>
  <card sat-history=”true” id=”C1”>
    <p> card 1 </p>
    <do type=”accept”>
      <go href=”#C2”/>
    </do>
  </card>

  <card sat-history=”false” id=”C2”>
    <p> card 2 </p>
    <do type=”accept”>
      <go href=”#C3”/>
    </do>
  </card>

  <card sat-history=”true” id=”C3”>
    <p> card 3 </p>
    <p> Please press back button </p>
    <do type=”accept”>
      <go href=”#C1”/>
    </do>
  </card>


</satml>
```

Result

If the back button is pressed within card 3 by the user (ie when the string “card 3” is displayed), card 1 but not card 2 should be reloaded (ie the string “card 1” should be displayed).

3.1.2.4 TEST_DECK_LEVEL_card.04

Description

Task is to test, if browser context is reset (ie if temporary variables are reset) when a card will be entered.

Source

```
<satml>
  <card id=”C1” newcontext=”false”>
    <p>card 1</p>
    <do type=”accept” label=”to card2”>
      <go href=”#C2”>
        <setvar name=”myVariable” value=”Hello World”/>
      </go>
    </do>
    <do type=”accept” label=”to card3”>
      <go href=”#C3”>
        <setvar name=”myVariable” value=”Hello World”/>
      </go>
    </do>
  </card>
  <card id=”C2” newcontext=”true”>
    <p>Value of my variable = $myVariable</p>
  </card>
<card id=”C3”>
<p> value of my variable = $myVariable</p>
</card>
</satml>
```

Result

Upon choosing to go to card2, the ME must display string "Value of my variable =", because variables which have not been initialised must by initialised by default with an empty string "". Upon choosing to go to card3, the ME must display “Value of my variable= Hello World”.

3.2 Variables

3.2.1 Empty variables

3.2.1.1 Test_temporaryVariables_empty.01

Description

Test empty variable, when variable is referenced without parenthesis.

Context : the variable “firstname” is not initialized before executing the deck.

Source

```
<wml>
    <card>
        <p>
            hello $firstname
        </p>
    </card>
</wml>
```

Result

“hello” is displayed.

3.2.1.2 Test_temporaryVariables_empty.02

Description :

Test empty variable, when variable is referenced with parenthesis.

The variable “firstname” is not initialized before executing the deck.

Source

```
<wml>
    <card>
        <p>
            hello $(firstname)
        </p>
    </card>
</wml>
```

Result

“hello” is displayed.

3.2.2 Initialised variables

3.2.2.1 Test_temporaryVariable_initialised.01

Description

Test variable initialisation when input is entered by the user and variable reference is made without parenthesis.

Source

```
<wml>
    <card>
        <p> enter your first name
        <input name="firstname"/> <br/>
            hello $firstname and bye !
        </p>
    </card>
</wml>
```

Result

The user is asked for his first name. He enters "simalliance". Then "hello simalliance and bye ! " is displayed.

3.2.2.2 Test_temporaryVariable_initialised.02

Description

Test variable initialisation when input is entered by the user and variable reference is made with parenthesis.

Source

```
<wml>
    <card>
        <p> enter your first name
        <input name="firstname"/> <br/>
            hello $(firstname) and bye !
        </p>
    </card>
</wml>
```

Result

same as for Test_temporaryVariable_initialised.01

3.2.2.3 Test_temporaryVariable_initialised.03

Description

Test variable initialisation when <setvar> tag is used.

Source

```
<wml>
    <card>
        <p>  the value of f is $(f)
            <anchor> go to myURL
                <go href=”#card2”>
                    <setvar name=”f” value=”simalliance”/>
                </go>
            </anchor>
        </p>
    </card>
    <card id=”card2”>
        <p>  the value of f is $(f)</p>
    </card>
</wml>
```

Result

The first card displays “the value of f is”. Once the user selects the hyperlink, the second card displays “the value of f is simalliance”.

3.2.2.4 Test_temporaryVariable_initialised.04

Description

Test variable initialisation when <setvar> tag is used, and the value to assign was entered by the user (mix of preceding cases).

Source

```
<wml>
    <card>
        <p>  the value of f is $(f) <br/>
        please enter the value for f
        <input name=”userchoice”/> <br/>
            <anchor> go to card2
                <go href=”#card2”>
                    <setvar name=”f” value=”$(userchoice)”/>
                </go>
            </anchor>
        </p>
    </card>
    <card id=”card2”>
        <p>  the value of f is $(f)</p>
    </card>
</wml>
```

Result

The text “the value of f is” is displayed. Then the text “enter the value for f” is displayed. The user enters “simalliance”. Once the user selects the hyperlink “go to card2” , the text “the value of f is simalliance” is displayed.

3.2.3 Using variables for VAS

3.2.3.1 Test_temporaryVariable_forVAS.01

Description

Test sending variables to a dynamic deck, with parameters in the URL name. In the example below, the string “hostname” has to be replaced by the IP address or host name of the HTTP server.

Source

```
<wml>
    <card>
        <p>  enter your first name <input name=”fname”/> <br/>
            enter your last name <input name=”lname”/> <br/>
            <a href=”http://hostname/myFile.cgi?f=$(fname)&amp;l=$(lname)”>go to myFile </a>
        </p>
    </card>
</wml>
```

Result

The user is asked for its firstname. He enters “sim”. Then he is asked for its last name, he enters “alliance”. Then the user selects the URL “go to my URL”. It has to be checked that the HTTP request sent by the gateway is :

GET http://hostname/myFile.cgi?f=sim&l=alliance

3.2.3.2 Test_temporaryVariable_forVAS.02

Description

Test sending variables to a dynamic deck, using the <postfield> tag. In the example below, the string “hostname” has to be replaced by the IP address or host name of the HTTP server.

Source

```
<wml>
```

```
    <card>
        <p>  enter your first name <input name=”fname”/> <br/>
            enter your last name <input name=”lname”/> <br/>
            <anchor> go to myURL
         <go href=”myURL.cgi”>
            <postfield name=”f” value=”$(fname)” />
            <postfield name=”l” value=”$(lname)” />
        </go>
        </anchor>
        </p>
    </card>
</wml>
```

Result

Same as for test_temporaryVariable_forVAS.01

3.2.3.3 Test_temporaryVariable_forVAS.03

Description

Test sending variables to a dynamic deck, using the <postfield> tag. The request sent is POST request. In the example below, the string “hostname” has to be replaced by the IP address or host name of the HTTP server.

Source

```
<wml>
    <card>
        <p>  enter your first name <input name=”fname”/> <br/>
            enter your last name <input name=”lname”/> <br/>
            <anchor> go to myURL
         <go href=”myURL.cgi” method=”post”>
            <postfield name=”f” value=”$(fname)” />
            <postfield name=”l” value=”$(lname)” />
        </go>
        </anchor>
        </p>
    </card>
</wml>
```

Result

The user is asked for its firstname. He enters “sim”. Then he is asked for its last name, he enters “alliance”. Then the user selects the URL “go to my URL”. It has to be checked that the HTTP request sent by the gateway is a POST request to : http://hostname/myFile.cgi, with parameters “f=sim” and “l=alliance”.

3.2.4 Using variables for attributes

3.2.4.1 Test_temporaryVariable_attributes_href.01

Description

Using a variable for the URL to be reached. The URL entered is valid.

Source

```
<wml>
    <card>
        <p>
        enter the URL you want to go to
        <input name=”userchoice”/> <br/>
            <a href=”$userchoice”> go !! </a>
        </p>
    </card>
</wml>
```

Result

The first card displays “enter the URL you want to go to”. The user enters a valid URL of a SATML/WML deck. Then he gets the first card of the requested deck.

3.2.4.2 Test_temporaryVariable_attributes_href.02

Description

Using a variables for the URL to be reached. The URL entered is not valid.

Source : same as for Test_temporaryVariable_attributes_href.01

Result

The first card displays “enter the URL you want to go to”. The user enters “http://noDeck.wml”, and he gets a deck from the gateway, which explains the requested deck could not be accessed.

3.2.4.3 Test_temporaryVariable_attributes_href.03

Description

Using a variable for the URL to be reached. The URL entered is valid.

Source

```
<wml>
    <card>
```

```
        <p>
        enter the URL you want to go to
        <input name=”userchoice”/> <br/>
            <do type=”accept” label=”go!”> <go href=”$userchoice”/> </do>
        </p>
    </card>
</wml>
```

Result

The same as for Test_temporaryVariable_attributes_href.01, except that the rendering on the screen may be different (because the rendering of “a” and “do” may be different).

3.2.4.4 Test_temporaryVariable_attributes_onpick.01

Description : Test_temporaryVariable_attributes_onpick.01

Source

```
<satml>
    <card>
        <p>
        enter the URL for the first option
        <input name=”userchoice”/> <br/>
            <select>
                <option onpick=”$userchoice”> your choice </option>
                <option onpick=”sim:/s/home”> home page </option>
            </select>
        </p>
    </card>
</satml>
```

Result :

The user is asked for entering a URL. He enters the URL. Then he has a “select”. If he chooses the first option, a request to the URL entered before is sent :

<u>Success case</u> : if the requested deck exists, then its first card is displayed

<u>Error case</u> : if the deck does not exist, then an error is displayed.

If the user chooses the second option, he goes to the home page.

3.2.4.5 Test_temporaryVariable_attributes_value.01

Description : Test_temporaryVariable_attributes_value.01

Context : we assume we have an handset supporting the feature “default value for input”.

Source

```
<wml>
    <card>
        <p>
            enter the default for the firstname <br/>
            <input name=”default”/> <br/>
            enter your firstname <br/>
            <input name=”userchoice” value=”$(default)”/> <br/>
            </p>
    </card>
</wml>
```

Result :

The text “enter the default for the firstname” is displayed. The user enters “simalliance”. Then the text “enter your firstname” is displayed, with “simalliance” as default value.

3.2.4.6 Test_temporaryVariable_attributes_help.01

Description

Context : the contextual menu “help” contains the default items.

Source

```
<satml>
    <card>
        <p>
            enter the text for the help <br/>
            <input name=”helptext”/> <br/>
            enter your firstname <br/>
            <input name=”userchoice” sat-help=”$helptext”/> <br/>
            </p>
    </card>
</satml>
```

Result

The text “enter the text for the help” is displayed. The user enters “the help string”. Then the text “enter your first name” is displayed. If the user presses on the “help” contextual menu, the text “the help string” is displayed.

3.2.4.7 Test_temporaryVariable_attributes_playtoneTitle.01

Description

Test title of “play tone” when it is a variable entered by the user.

Source

```
<satml>
    <card>
        <p>
            enter the text for the title <br/>
            <input name=”title”/> <br/>
        </p>
            <sat-play-tone sat-title=”$title”/>
    </card>
</satml>
```

Result

The text “enter the text for the title” is displayed. The user enters “the great title”. Then a tone is played, with “the great title” as title for the “play-tone”.

3.2.4.8 Test_temporaryVariable_attributes_setupCallConfirm.01

Description

Test the confirmation message of “setupcall” when this message is entered by the user.

Source

```
<satml>
    <card>
        <p>
            enter the confirm text <br/>
            <input name=”confirm”/> <br/>
            </p>
        <sat-setup-call
            sat-dest=”+33147466667”
            sat-confirm=”$confirm”/>
    </card>
</satml>
```

Result

The text “enter the confirm text” is displayed. The user enters “hello”. Then the call is asked for confirming the call up with the text “hello”.

3.2.4.9 Test_temporaryVariable_attributes_LInfoURL.01

Description

Test variable use in the URL reference to go in <sat-local-info>.

Source

```
<satml>

    <card>

        <p>

            enter the URL <br/>

            <input name=”url”/> <br/>

            </p>

        <sat-local-info

            sat-name=”li”

            sat-href=”$url”/>

    </card>

</satml>
```

Result

The text “enter the URL” is displayed. The user enters a valid url. The gateway sends the request:
GET theURL?li=someHexaString

In this test “someHexaString” represents the hexa string resulting from the STK “provide local info” command, and “theURL” represents the URL entered by the user.

3.2.5 Use of special charcaters in variables

3.2.5.1 Test_temporaryVariable_special.01

Description

Test if display of “dollar” works.

Source

```
<wml>

    <card>

        <p>

            Here is a dollar : $$

        </p>

    </card>
```

```
</wml>
```

Result

The text : “ Here is a dollar : $ ” is displayed.

3.2.5.2 Test_temporaryVariable_special.02

Description

Aim : test escaping and unescaping

Source

```
<wml>
    <card>
        <p>
        enter the value for x : <br/>
            <input name=”x”/>
        x escaped is : $(x:e) <br/>
        x unescaped is $(x:u) <br/>
        x noescaped is $(x:n) <br/>
        </p>
    </card>
</wml>
```

Result

The text “enter the value for x” is displayed. The user enters “simalliance”. Then the text below is displayed :

“x escaped is simalliance

x unescaped is simalliance

x noescaped is simalliance”.

3.3 Fields

3.3.1 <input>

3.3.1.1 TEST_CONTROL_INPUT.01

Description

Simple input field with no parameters.

Source

```
<wml>
  <card id="tinp01">
    <p>
```

```
      input: <input name="input"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall read any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard until the (user agent dependant) default maximum length is reached. It shall then display the given input e.g. "*input value is: ABCdef123(/&%)@äé*".

3.3.1.2 TEST_CONTROL_INPUT.02

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp02" >
    <p>
      input: <input name="input" format="1M"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall read any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard. Only one character shall be read. It shall then display the given input e.g. "*input value is: 3*".

3.3.1.3 TEST_CONTROL_INPUT.03

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp03" >
    <p>
      input: <input name="input" format="MMMM"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

As in test case 3.3.1.1, SB shall prompt for an input by displaying "*input:*" and then shall read any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard until the (user agent dependant) default maximum length is reached. It shall then display the given input e.g. "*input value is: A2d6*".

3.3.1.4 TEST_CONTROL_INPUT.04

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp04" >
    <p>
      input: <input name="input" format="4M"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall read any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard. Four characters must be entered. It shall then display the given input e.g. "*input value is: A2d6*".

3.3.1.5 TEST_CONTROL_INPUT.05

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp05" >
    <p>
      input: <input name="input" format="*M" emptyok="true"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall read any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard until the (user agent dependant) default maximum length is reached. The input field may also be left empty. It shall then display the given input e.g. "*input value is:* ".

3.3.1.6 TEST_CONTROL_INPUT.06

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp06" >
    <p>
      input: <input name="input" format="*N"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall read only numeric characters (including "+", "*", "#") at the keyboard until the (user agent dependant) default maximum length is reached. It shall then display the given input e.g. "*input value is: +1234567890*".

3.3.1.7 TEST_CONTROL_INPUT.07

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp07" >
    <p>
     input: <input name="input" format="1N"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall read only numeric characters (including "+", "*", "#") at the keyboard. Only one character shall be read. It shall then display the given input e.g. "*input value is: 1*".

3.3.1.8 TEST_CONTROL_INPUT.08

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp08" >
    <p>
      input: <input name="input" format="*N" emptyok="true"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall read only numeric characters (including "+", "*", "#") until the (user agent dependant) default maximum length is reached. The input field may also be left empty. It shall then display the given input e.g. "*input value is:* ".

3.3.1.9 TEST_CONTROL_INPUT.09

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp09" >
    <p>
      input: <input name="input" format="*A"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

As in test case 3.3.1.3

3.3.1.10 TEST_CONTROL_INPUT.10

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp10" >
    <p>
      input: <input name="input" format="*n"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

As in test case 3.3.1.3

3.3.1.11 TEST_CONTROL_INPUT.11

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp11" >
    <p>
      input: <input name="input" maxlength="15"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall accept up to 15 symbolic, numeric, uppercase or lowercase alphabetic characters at the keyboard. It shall then display the given input e.g. "*input value is: 1234567890abcde*". At least one char must be entered.

3.3.1.12 TEST_CONTROL_INPUT.12

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp12" >
```

```
    <p>
      input: <input name="input" maxlength="15" emptyok="true"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall accept up to 15 symbolic, numeric, uppercase or lowercase alphabetic characters at the keyboard. It shall then display the given input e.g. "*input value is: 1234567890abcde*". The field may be left empty.

3.3.1.13 TEST_CONTROL_INPUT.13

Description

Input field with format parameters.

Source

```
<wml>
  <card id="tinp13" >
    <p>
      input: <input name="input" size="15"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall read any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard until the default (user agent dependant) maximum length is reached. It shall then display the given input e.g. "*input value is: ABCdef123(/&%)@*". The size element shall be ignored by the DE.

3.3.1.14 TEST_CONTROL_INPUT.14

Description

Input field with format parameters.

Source

```
<satml>
  <card id="tinp14" >
    <p>
      input: <input name="input" sat-minlength="5"/>
      input value is: $input
    </p>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall read any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard until the default (user agent dependant) maximum length is reached. It shall then display the given input e.g. "*input value is: ABCdef123(/&%)@*". At minimum 5 characters must be entered.

3.3.1.15 TEST_CONTROL_INPUT.15

Description

Input field with format parameters.

Source

```
<satml>
  <card id="tinp15" >
    <p>
      input: <input name="input" sat-minlength="300"/>
      input value is: $input
    </p>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying “*input:*” and then shall read any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard until the default (user agent dependant) maximum length is reached. At minimum 300 characters are requested to be entered, this should be mapped to a value allowed by GSM 11.14 by DE. Check that no undefined behaviour of the DE or SB occurs.

3.3.1.16 TEST_CONTROL_INPUT.16

Description

Input field with format parameters.

Source

```
<satml>
  <card id="tinp16" >
    <p>
      input: <input name="input" emptyok="true" sat-minlength="5"/>
      input value is: $input
    </p>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying “*input:*” and then shall read any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard until the default (user agent dependant) maximum length is reached. It shall then display the given input e.g. “*input value is: ABCdef123(/&%)@*”. At minimum 5 characters must be entered. The emptyok element shall be ignored by DE.

3.3.1.17 TEST_CONTROL_INPUT.17

Description

Simple input field with title parameter.

Source

```
<wml>
   <card id="tinp17">
    <p>
      <input name="input" title="input:"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

As in test case 3.3.1.3

3.3.1.18 TEST_CONTROL_INPUT.18

Description

Simple input field with title parameter defined by variable.

Source

```
<wml>
  <card id="tinp18">
    <p>
      <input name="mytit" title="title:"/>
      <input name="input" title="$(mytit):"/>
      input value is: $input
    </p>
  </card>
</wml>
```

Result

SB shall prompt for an input by displaying "*title:*" and then shall read any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard. It shall then prompt again for an input by displaying the entered string, e.g. *"Input:"*. After reading any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard it shall then display the given input e.g. "*input value is: A2d6*".

3.3.1.19 TEST_CONTROL_INPUT.19

Description

Input field with format parameters and help string.

Source

```
<satml>
  <card id="tinp19" >
    <p>
      input: <input name="input" format="5N"
                    sat-help="Enter a 5 digit number."/>
      input value is: $input
    </p>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall accept exact five numeric characters (including "+", "*", "#"). When the user presses the help button while asked for input it shall display the text "Enter a 5 digit number" assuming that the contextual menu is in default state. After leaving the help display the prompt for input shall be re-displayed. When five characters have been entered the SB shall display the given input e.g. "*input value is: 12345*".

3.3.1.20 TEST_CONTROL_INPUT.20

Description

Input field with format parameters and help strings.

Source

```
<satml sat-help="Look into the Browser manual." >
  <card id="tinp20" >
```

```
    <p>
      input: <input name="input" format="5N"
                    sat-help="Enter a 5 digit number."/>
      input value is: $input
    </p>
  </card>
</satml>
```

Result

As in test case 3.3.1.19

3.3.1.21 TEST_CONTROL_INPUT.21

Description

Input field with a password type.

Source

```
      <satml>
  <card id="tinp22"
        sat-help="Fill in the input field.">
    <p>
      input: <input name="input" type="password"
                    emptyok="true" sat-minlength="2" maxlength="5"/>
      input value is: $input
    </p>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*input:*" and then shall accept from 2 up to 5 numeric characters. The entered digits must not be visible. Once the characters have been entered, the SB shall display the given input e.g. "*input value is: 1234*".

3.3.2 <select>

3.3.2.1 TEST_CONTROL_SELECT.01

Description

Select element with two options and no title but preceding text to be used as select prompt instead.

Source

```
<wml>
  <card id="tsel01" >
    <p>
      Please make your selection:
      <select name=”choice”>
        <option value=”item 1”>Option 1</option>
        <option value=”item 2”>Option 2</option>
      </select>
      Your selection was: $(choice).
    </p>
  </card>
</wml>
```

Result

The result depends much on the layout of the ME's display. The SB shall display the text "*Please make your selection:*". In the choice menu, the options "*Option 1*" and "*Option 2*" shall be displayed. The user may toggle the graphical selection of these items as long as the OK button is not pressed. When the OK button is pressed the DE shall display the text "*Your selection was: item 1.*" when the last selected item was Option 1 before pressing the OK button else the text "*Your selection was: item 2.*"

3.3.2.2 TEST_CONTROL_SELECT.02

Description

Select element with two options and title.

Source

```
<wml>
  <card id="tsel02" >
    <p>
      <select name=”choice” title=”Please make your selection:”>
        <option value=”item 1”>Option 1</option>
        <option value=”item 2”>Option 2</option>
      </select>
      Your selection was: $(choice).
    </p>
  </card>
</wml>
```

Result

As in test case 3.3.2.1.

3.3.2.3 TEST_CONTROL_SELECT.03

Description

Select element with two options and title as well as preceding text.

Source

```
<wml>
  <card id="tsel03" >
    <p>
      You can choose among two options.
      <select name=”choice” title=”Please make your selection:”>
        <option value=”item 1”>Option 1</option>
        <option value=”item 2”>Option 2</option>
      </select>
      Your selection was: $(choice).
    </p>
  </card>
</wml>
```

Result

First the SB shall display the string "*You can choose among two options.*" Then it shall proceed as in test case 3.3.2.1.

3.3.2.4 TEST_CONTROL_SELECT.04

Description

Select element with no title but preceding text to be used as prompt, options and help text.

Source

```
<satml>
  <card id="tsel04" >
    <p>
      Please make your selection:
      <select name=”choice”>
        <option value=”item 1” sat-help=”Item 1 is the first element.”>
Option 1
</option>
        <option value=”item 2” sat-help=”Item 2 is the second element.”>
Option 2
</option>
      </select>
      Your selection was: $(choice).
    </p>
  </card>
</satml>
```

Result

The DE shall display the text *“Please make your selection:”*. In the choice menu, the options *“Option 1”* and *“Option 2”* shall be displayed. The user may toggle the graphical selection of these items as long as the OK button is not pressed. When pressing the help button the help text *”Item 1 is the first element.”* or *”Item 2 is the second element.”* shall be displayed, depending on the selected option and assuming that the contextual menu is in default state. After leaving the help display the select list shall be re-displayed. When pressing the OK button the DE shall display the text *“Your selection was: item 1.”* when the last selected item was Option 1 before pressing the OK button else the text *“Your selection was: item 2.”*

3.3.2.5 TEST_CONTROL_SELECT.05

Description

Select element with two options and title, using the iname attribute.

Source

```
<wml>
  <card id="tsel05" >
    <p>
      <select iname=”index” title=”Please make your selection:”>
        <option value=”item 1”>Option 1</option>
        <option value=”item 2”>Option 2</option>
      </select>
      Your selection was: item $(index).
    </p>
  </card>
</wml>
```

Result

If the first option was selected, the text is “Your selection was 1”. Otherwise it is “Your selection was 2”.

3.3.2.6 TEST_CONTROL_SELECT.06

Description

Select element with two options and title, using the name and iname attributes.

Source

```
<wml>
  <card id="tsel06" >
    <p>
      <select name=”choice” iname=”index” title=”Please make your selection:”>
```

```
        <option value=”item 1”>Option 1</option>
        <option value=”item 2”>Option 2</option>
      </select>
      Your selection was: $(choice) at position $(index).
    </p>
  </card>
</wml>
```

Result

As in test case 3.3.2.1, however, the displayed result is *“Your selection was: item 1 at position 1.”* when the last selected item was Option 1 before pressing the OK button else the text *“Your selection was: item 2 at position 2.”*

3.3.2.7 TEST_CONTROL_SELECT.07

Description

Select element with a long option list and title.

Source

```
<wml>
  <card id="tsel07" >
    <p>
      <select name=”choice” title=”Please make your selection...”>
        <option value=”item 01”>The option #01.</option>
        <option value=”item 02”>The option #02.</option>
        <option value=”item 03”>The option #03.</option>
        <option value=”item 04”>The option #04.</option>
        <option value=”item 05”>The option #05.</option>
        <option value=”item 06”>The option #06.</option>
        <option value=”item 07”>The option #07.</option>
        <option value=”item 08”>The option #08.</option>
        <option value=”item 09”>The option #09.</option>
        <option value=”item 10”>The option #10.</option>
        <option value=”item 11”>The option #11.</option>
        <option value=”item 12”>The option #12.</option>
      </select>
      Your selection was: $(choice).
    </p>
  </card>
</wml>
```

Result

In this long option list more than 256 bytes would be required for the SELECT ITEM command of GSM 11.14. The DE may split up the option list into multiple select statements, but this is no mandatory feature. Check that no undefined behaviour of the DE or SB occurs.

3.3.2.8 TEST_CONTROL_SELECT.08

Description

Select element with two options and title, using a variable in the option text.

Source

```
<wml>
  <card id="tsel08" >
    <p>
      <input name=”myopt” title=”Enter your own option:”/>
      <select name=”choice” title=”Please make your selection:”>
        <option value=”item 1”>Option 1</option>
        <option value=”item 2”>$myopt</option>
      </select>
      Your selection was: $(choice).
    </p>
  </card>
</wml>
```

Result

The SB shall display the text ”*Enter your own option:* ” and read a string of any characters. Then, the result depends much on the layout of the ME’s display. The SB shall display the text “*Please make your selection:*” at the top of a choice menu. Below the options “*Option 1*” and the one with the user-defined text, e.g. ”*my option*”, shall be displayed. The user may toggle the graphical selection of these items as long as the OK button is not pressed. When the OK button is pressed the DE shall display the text “*Your selection was: item 1.*” when the last selected item was Option 1 before pressing the OK button else the text “*Your selection was: item 2.*”

3.3.2.9 TEST_CONTROL_SELECT.09

Description

Select element with two options and title, using a variable in the title text.

Source

```
<wml>
  <card id="tsel09" >
    <p>
      <input name=”mytit” title=”Enter your own title:”/>
      <select name=”choice” title=”$(mytit):”>
        <option value=”item 1”>Option 1</option>
        <option value=”item 2”>Option 2</option>
      </select>
      Your selection was: $(choice).
    </p>
  </card>
</wml>
```

Result

The SB shall display the text ”*Enter your own title:* ” and read a string of any characters. Then, the result depends much on the layout of the ME’s display. The SB shall display the entered title text, e.g. “*Choice:*” at the top of a choice menu. Below the options “*Option 1*” and ”*Option 2*”, shall be displayed. The user may toggle the graphical selection of these items as long as the OK button is not pressed. When the OK button is pressed the DE shall display the text “*Your selection was: item 1.*” when the last selected item was Option 1 before pressing the OK button else the text “*Your selection was: item 2.*”

3.3.2.10 TEST_CONTROL_SELECT.10

Description

Select element with a long option list and title, using a variable.

Source

```
<wml>
  <card id="tsel10" >
    <p>
      <input name=”mytit” title=”Enter your own title:”/>
      <select name=”choice” title=”$(mytit):”>
        <option value=”item 01”>The option #01.</option>
        <option value=”item 02”>The option #02.</option>
        <option value=”item 03”>The option #03.</option>
        <option value=”item 04”>The option #04.</option>
        <option value=”item 05”>The option #05.</option>
        <option value=”item 06”>The option #06.</option>
        <option value=”item 07”>The option #07.</option>
        <option value=”item 08”>The option #08.</option>
        <option value=”item 09”>The option #09.</option>
        <option value=”item 10”>The option #10.</option>
        <option value=”item 11”>The option #11.</option>
        <option value=”item 12”>The option #12.</option>
      </select>
      Your selection was: $(choice).
    </p>
  </card>
</wml>
```

Result

The SB shall display the text ”*Enter your own title:* ” and read a string of any characters. Then, the result depends on the length of the user input. If it is longer than 28 characters then at least 256 bytes would be required for the SELECT ITEM command of GSM 11.14. The DE cannot predict the length of the title, this can only detected by SB. Check that no undefined behaviour of the DE or SB occurs. If the user input is not longer than 28 characters the SB shall process the select command as in test case 3.3.2.1, however, showing a list of twelve options each of them like *"The option #01."*

3.3.2.11 TEST_CONTROL_SELECT.11

Description

Select element with navigation options using the onpick attribute. This test will require the existence of two more WML / SATML pages with the name “option1” and “option2” in the same directory.

Source

```
<wml>
  <card id="tsel11" >
    <p>
      Where do you want to go?
      <select>
        <option onpick=”http://myURL/myFile1.wml”>Option 1</option>
        <option onpick=”http://myURL/myFile2.wml”>Option 2</option>
      </select>
    </p>
  </card>
</wml>
```

Source of myFile1.wml

```
<wml>
  <card>
    <p>
      You have gone to Option 1.
    </p>
   </card>
</wml>
```

Source of myFile2.wml

```
<wml>
  <card>
    <p>
      You have gone to Option 2.
    </p>
```

```
   </card>
</wml>
```

Result

The SB shall display the text "*Where do you want to go?*". In the choice menu, the options "*Option 1*" and "*Option 2*" shall be displayed. The user may toggle the graphical selection of these items as long as the OK button is not pressed. When pressing the OK button the DE shall display the text "*You have gone to Option 1.*" when the last selected item was Option 1 before pressing the OK button else the text "*You have gone to Option 2.*"

3.3.2.12 TEST_CONTROL_SELECT.12

Description

Select element with navigation options using an onevent element. This test will require the existence of two more WML / SATML pages with the name "option1" and "option2" in the same directory.

Source

```
<wml>
  <card id="tsel12" >
    <p>
      Where do you want to go?
      <select>
        <option>
          <onevent type="onpick">
            <go href=”option1” sendreferer="true"/>
          </onevent>
          Option 1
        </option>
        <option onpick=”option2”>Option 2</option>
      </select>
    </p>
  </card>
</wml>
```

Source of option1:

```
<wml>
  <card>
    <p>
      You have gone to Option 1.
    </p>
   </card>
</wml>
```

Source of option2:

```
<wml>
  <card>
    <p>
      You have gone to Option 2.
    </p>
   </card>
</wml>
```

Result

The SB shall display the text "*Where do you want to go?*" at the top of a choice menu. Below the options "*Option 1*" and "*Option 2*" shall be displayed. The user may toggle the graphical selection of these items as long as the OK button is not pressed. When pressing the OK button the DE shall display the text "*You have gone to Option 1.*" when the last selected item was Option 1 before pressing the OK button else the text "*You have gone to Option 2.*" In the first case the referring URI address shall be sent to the origin server, in the second case not.

3.3.2.13 TEST_CONTROL_SELECT.13

Description

Select element with mixed value assignment and navigation options. This test will require the existence of two more WML / SATML pages with the name "option1" and "option2" in the same directory.

Source

```
<wml>
  <card id="tsel13" >
    <p>
      What do you want to choose?
      <select name="choice">
        <option value="item 1" onpick="option1">Option 1</option>
        <option onpick="option2">Option 2</option>
        <option value="item 3">Option 3</option>
      </select>
      Your selection was: $(choice).
    </p>
  </card>
</wml>
```

Source of option1:

```
<wml>
  <card>
    <p>
      You have gone to Option 1.<br>
      Your selection was: $choice.
    </p>
  </card>
</wml>
```

Source of option2:

```
<wml>
  <card>
    <p>
      You have gone to Option 2.<br>
      Your selection was: $(choice).
    </p>
  </card>
</wml>
```

Result

In the current SATML version support of selects with mixed assignment and navigation options is an optional feature. If it is supported the SB shall display the text "*Where do you want to go?*". In the choice menu, the options "*Option 1*", "*Option 2*", and "*Option 3*" shall be displayed. The user may toggle the graphical selection of these items as long as the OK button is not pressed. When pressing the OK button the DE shall display the text "*You have gone to Option 1.*" and "*Your selection was: item 1.*"when the last selected item was Option 1 before pressing the OK button. Else the text "*You have gone to Option 2.*" and "*Your selection was: .*" shall be displayed when the last selected item was Option 2 before pressing the OK button. Else the text "*Your selection was: item 3.*" shall be displayed. If the feature is not supported check that no undefined behaviour of the DE or SB occurs.

3.3.2.14 TEST_CONTROL_SELECT.14

Description

Select element with a hierarchical option list using optgroups.

Source

```
<wml>
  <card id="tsel14" >
    <p>
      <select name=”choice” title=”Please make your selection:”>
        <optgroup title=”01-03” >
          <option value=”item 01”>The option #01.</option>
          <option value=”item 02”>The option #02.</option>
          <option value=”item 03”>The option #03.</option>
        </optgroup>
        <optgroup title=”04-06” >
          <option value=”item 04”>The option #04.</option>
          <option value=”item 05”>The option #05.</option>
          <option value=”item 06”>The option #06.</option>
        </optgroup>
        <optgroup title=”07-12” >
          <optgroup title=”07-09” >
            <option value=”item 07”>The option #07.</option>
            <option value=”item 08”>The option #08.</option>
            <option value=”item 09”>The option #09.</option>
          </optgroup>
          <option value=”item 10”>The option #10.</option>
          <option value=”item 11”>The option #11.</option>
          <option value=”item 12”>The option #12.</option>
        </optgroup>
      </select>
      Your selection was: $(choice).
    </p>
  </card>
</wml>
```

Result

The result depends much on the layout of the ME’s display and whether the DE chooses to make use of optgroup elements for an hierarchical presentation of selections. The behaviour of the SB is similar to test case 3.3.2.1, however, showing a list or a hierarchy of twelve options each of them like *"The option #01."*

3.4 Contents

3.4.1 <p>

3.4.1.1 TEST_TEXT_P.01

Description

Test text display.

Source :

```
<wml>

  <card>

    <p> hello world </p>

  </card>

</wml>
```

Result

The text “hello world” is displayed.

3.4.1.2 TEST_TEXT_P.02

Description

Test text display when it is several paragraphs in the WML source.

Source

```
<wml>
    <card>
        <p> hello </p>
        <p> world </p>
    </card>
</wml>
```

Result :

The text “hello” is displayed, and, after user has pressed “ok”, “world” is displayed.

3.4.1.3 TEST_TEXT_P.03

Description

Test text display with normal priority.

Source

```
<satml>
    <card>
        <p sat-prio=”normal”> hello world </p>
    </card>
</satml>
```

Result :

Same as for TEST_TEXT_P.01.

3.4.1.4 TEST_TEXT_P.04

Description

Test text display with “auto clear”.

Source :

```
<satml>
    <card>
```

```
        <p sat-auto-clr=”true”> hello world </p>
        <p> bye </p>
    </card>
</satml>
```

Result :

The text “hello world” is displayed, and, without any user interaction, “bye” is displayed.

3.4.1.5 TEST_TEXT_P.05

Description

Test text before an <input> tag.

Source :

```
<satml>
    <card>
        <p sat-auto-clr=”true”> enter your age <input name=”age”/> </p>
    </card>
</satml>
```

Result :

The text “enter your age” is displayed, and then the user is asked for input.

3.4.1.6 TEST_TEXT_P.06

Description

Test display of a long text.

Source

```
<wml>
    <card>
        <p> this is an example with a very long text : sentence 1
        this is an example with a very long text : sentence 2
        this is an example with a very long text : sentence 3
        this is an example with a very long text : sentence 4
        this is an example with a very long text : sentence 5
        this is an example with a very long text : sentence 6
        this is an example with a very long text : sentence 7
        </p>
    </card>
```

```
</wml>
```

Result

There are two possible outputs for this :

- Either the entire text is displayed, with maybe user interactions ("OK") between sentences
- Or an error message is displayed on the browser.

3.4.1.7 TEST_TEXT_P.07

Description

Test if formatting tag are properly ignored.

Source

```
<wml>
   <card>
      <p> <em> hello <strong> world </strong></em> <b> and </b> <i>welcome </i> <u> to </u>
      <big> SAT</big> <small> bye bye </small>
      </p>
   </card>
</wml>
```

Result :

The text "hello world and welcome to SAT bye bye" is displayed.

3.4.1.8 TEST_TEXT_P.08

Description

Test <img> tag.

Source

```
<wml>
   <card>
      <p> hello <img alt="an image" src="http://mysite/myimage.gif"/> </p>
   </card>
</wml>
```

Result

The text "hello" or "hello an image" is displayed.

3.4.1.9 TEST_TEXT_P.09

Description

Test that the <table> tag and other tag related to table management are properly ignored.

Source

```
<wml>
   <card>
      <p> hello
      <table columns="2">
         <tr> <td>hello </td> <td> world </td></tr>
         <tr> <td> good </td> <td> bye </td> </tr>
      </table>
      </p>
   </card>
</wml>
```

Result :

The text "hello", "hello world", and "good bye" is displayed.

3.4.2 <br/>

3.4.2.1 TEST_TEXT_BR.01

Description

Test the <br/> tag

Source

```
<wml>
   <card>
      <p> hello <br/> world </p>
   </card>
</wml>
```

Result

The text "hello" is displayed, and, after user interaction or on another line, "world" is displayed.

3.5 Navigation

3.6 Navigation scenarios

3.6.1 From a resident deck

Context : for every test in this part, the S@TML/WML deck source must be encoded in SBC and stored as a resident deck in the SIM.

3.6.1.1 Test_navigation_resident.01

Description

test navigation from card to card in the same deck, where the requested card exists.

Source

```
<wml>
    <card>
        <p>
            resident tests n 1 <br/>
            <a href=”#card2”> go to card2 </a> <br/>
            <a href=”#card3”> go to card3 </a> <br/>
        </p>
    </card>
    <card id=”card2”>
        <p> welcome on card2 </p>
    </card>
</wml>
```

Result :

When the user selects the hyperlink “go to card2”, “welcome on card2” is displayed.

3.6.1.2 Test_navigation_resident.02

Description

test navigation from card to card in the same deck, where the requested card doesn’t exist.

Source : same as for Test_navigation_resident.01

Result :

When the user selects the hyperlink “go to card 3”, an error message is displayed.

3.6.1.3 Test_navigation_resident.03

Description

Test navigation from resident deck to online deck, where the online deck exists and can be converted into SBC.

In the test below, “myHost” must be replaced by the IP address or host name where the HTTp server is installed. The file "myFile.wml" contains a card called “tech”, but no card called “t”.

Source

```
<wml>
    <card>
```

```
        <p> resident tests 3, 4, 5 and 6 <br/>
            <a href=”http://myHost/myFile.wml”> go to myFile </a> <br/>
            <a href=”http://myHost/badFile.wml”> go to not existing URL </a> <br/>
            <a href=” #tech”> go to “tech” card in “myFile”</a> <br/>
            <a href=”http://myHost/myFile.wml #t”> go to “t” card in “myFile”</a> <br/>
        </p>
    </card>
</wml>
```

Result

When the user selects the hyperlink “go to myFile”, a request goes online and the first card of the requested deck is executed.

3.6.1.4 Test_navigation_resident.04

Description

Aim : test navigation from resident deck to online deck, where the online deck does not exist.

Source : same as for Test_navigation_resident.03

Result

When the user selects the hyperlink “go to not existing URL”, a request goes online but he gets a SBC deck indicating the deck can’t be reached.

3.6.1.5 Test_navigation_resident.05

Description

Test navigation from resident deck to online deck + card, where the requested card exists.

Source : same as for Test_navigation_resident.03

Result :

When the user selects the hyperlink “go to “tech” card in myFile”, a request goes online and the card “tech” is executed.

3.6.1.6 Test_navigation_resident.06

Description

Aim : test navigation from resident deck to online deck + card, where the requested card does not exist.

Source : same as for Test_navigation_resident.03

Result :

When the user selects the hyperlink “go to “t” card in myFile”, a request goes online and the deck “myFile” is downloaded. The browser generates an error indicating the card can’t be reached.

3.6.1.7 Test_navigation_resident.07

Description

Aim : test navigation from resident deck to resident deck.

Context : A resident deck called “deck1” is stored on the SIM, as a “resident deck”.

Source

```
<wml>
    <card>
        <p> resident tests 7, 8, 9 and 10 <br/>
            <a href=”sim:deck1”> go to deck1 </a> <br/>
            <a href=”sim:deck2”> go to deck2  </a> <br/>
            <a href=”sim:deck1#card2”> go to card2 in deck1 </a> <br/>
            <a href=”sim:deck1#card3”> go to card3 in deck1</a> <br/>
        </p>
    </card>
</wml>
```

Result :

When the user selects the hyperlink “go to deck1”, the first card of “deck1” is executed.

3.6.1.8 Test_navigation_resident.08

Description

Aim : test navigation from resident deck to resident deck. The requested deck does not exist.

Context : No resident deck called “deck2” exists on the SIM.

Source : same as for Test_navigation_resident.07

Result :

When the user selects the hyperlink “go to deck2”, an error message is displayed.

3.6.1.9 Test_navigation_resident.09

Description

Aim : test navigation from resident deck to resident deck + card, where the requested card exists.

Context : a resident deck called “deck1” exists on the SIM, and it contains a card called “card2”.

Source : same as for Test_navigation_resident.07

Result :

When the user selects the hyperlink “go to card2 on deck 1”, the requested card is executed.

3.6.1.10 Test_navigation_resident.10

Description

Aim : test navigation from resident deck to resident deck + card, where the requested card doesn't exist.

Context : a resident deck called "deck1" exists on the SIM. It doesn't contain any card called "card3".

Source : same as for Test_navigation_resident.07

Result :

When the user selects the hyperlink "go to card3 on deck1", an error message is displayed.

3.6.1.11 Test_navigation_resident.11

Description

Test navigation from resident deck to the home deck.

Source

```
<satml>
    <card>
        <p>
            resident test 11 and 12 <br/>
            <a href="sim:/s/home"> go to the home deck </a>
        </p>
    </card>
</satml>
```

Result :

When the user selects the hyperlink "go to the home deck", the home deck is executed.

3.6.2 From an online deck

3.6.2.1 Test_navigation_online.01

Description

Aim : test navigation from online deck to another card in the same deck, where the card exists.

Source

```
<wml>
    <card>
        <p>
            online tests  1 and 2 <br/>
            <a href="#card2"> go to card2 </a> <br/>
            <a href=”#card3”> go to card 3 </a> <br/>
        </p>
    </card>
    <card id=”card2”>
        <p> welcome on card2 </p>
    </card>
</wml>
```

Result :

When the user selects the hyperlink “go to card2”, “welcome on card2” is displayed.

3.6.2.2 Test_navigation_online.02

Description

Aim : test navigation from online deck to another card in the same deck, where the requested card doesn’t exist.

Source : same as for Test_navigation_online.01

Result :

When the user selects the hyperlink “go to card 3”, an error message is displayed.

3.6.2.3 Test_navigation_online.03

Description

Aim : test navigation from online deck to online deck.

Source

```
<wml>
    <card>
        <p> online tests 3, 4, 5 and 6 <br/>
            <a href=”http://myHost/myFile.wml”> go to myFile </a> <br/>
            <a href=”http://myHost/badFile.wml”> go to not existing URL </a> <br/>
            <a href=” http://myHost/myFile.wml #tech”> go to “tech” card in myFile </a> <br/>
            <a href=”http://myHost/myFile.wml #t”> go to “t” card in myFile </a> <br/>
        </p>
    </card>
</wml>
```

Result

When the user selects the hyperlink “go the myFile”, a request goes online and he gets the expected deck.

3.6.2.4 Test_navigation_online.04

Description : test navigation from online deck to online deck, where the requested deck does not exist.

Source : same as for Test_navigation_online.03

Result :

When the user selects the hyperlink “go to not existing URL”, a request goes online but he gets a SBC deck indicating the deck can’t be reached.

3.6.2.5 Test_navigation_online.05

Description : test navigation from online deck to online deck + card, where the requested card exists.

Source : same as for Test_navigation_online.03

Result :

When the user selects the hyperlink “go to “tech” card in myFile”, a request goes online and he gets the expected card.

3.6.2.6 Test_navigation_online.06

Description : test navigation from online deck to online deck + card, where the requested card doesn’t exist.

Source : same as for Test_navigation_online.03

Result :

When the user selects the “go to unexisting card”, a request goes online. The deck is downloaded and the browser generates an error explaining it can’t find the card.

3.6.2.7 Test_navigation_online.07

Description

Test navigation from online deck to resident deck, where the resident exists. A resident deck called “deck1” is stored as a resident deck on the SIM.

Source

```
<wml>
    <card>
        <p> online tests 7, 8, 9 and 10 <br/>
            <a href=”sim:deck1”> go to deck1 </a> <br/>
            <a href=”sim:deck2”> go to deck2  </a> <br/>
            <a href=”sim:deck1#card2”> go to card2 in deck1 </a> <br/>
            <a href=”sim:deck1#card3”> go to card3 in deck1</a> <br/>
        </p>
    </card>
</wml>
```

Result :

When the user selects the hyperlink “go to deck1”, the first card of “deck1” is executed.

3.6.2.8 Test_navigation_online.08

Description :test navigation from online deck to resident deck, where the resident deck doesn’t exist. No resident deck called “deck2” is stored on the SIM.

Source : same as for Test_navigation_online.07

Result :

When the user selects the hyperlink “go to deck2”, an error message is displayed.

3.6.2.9 Test_navigation_online.09

Description : test navigation from online deck to resident deck + card, where the card exists.

Source : same as for Test_navigation_online.07

Result :

When the user selects the hyperlink “go to card2 on deck 1”, the requested card is executed.

3.6.2.10 Test_navigation_online.10

Description : test navigation from online deck to resident deck + card, where the card doesn’t exist. A resident deck called “deck1” exist on the SIM, but doesn’t contain a card called “card3”.

Source : same as for Test_navigation_online.07

Result :

When the user selects the hyperlink “go to card3 on deck1”, an explicit error message is displayed by the browser.

3.6.2.11 Test_navigation_online.11

Description :_test navigation from online deck to the home deck. A home deck exists on the SIM.

Source

```
<satml>

    <card>

        <p>

            online test 11 and 12 <br/>

            <a href=”sim:/s/home”> go to the home deck </a>

        </p>

    </card>

</satml>
```

Result :

When the user selects the hyperlink “go to the home deck”, the home deck is executed.

4 Security features

4.1 Untrusted source

4.1.1 <sat-setup-call>

4.1.1.1 test_security_untrusted_setupcall.01

Description

It is assumed that the deck has been received from a not trusted source.

Source

```
<wml>
    <card>
        <p> hello </p>
        <sat-setup-call sat-dest=”+33147466667”/>
        <p> world </p>
    </card>
</wml>
```

Result :

Because the use of all S@TML STK Extension elements is in general restricted to trusted decks, the ME must not perform any attempt to set up a call. Either the STK Extension command SETUP CALL is ignored or an error message is displayed.

5 Telephony

In the all decks below, the phone number must be changed to a valid phone number.

5.1 WTAI functions

5.1.1 MakeCall

5.1.1.1 TEST_TELEPHONY_WTAI_MakeCall.01

Description

Test WTAI fucntion “make call” from WTAI public library.

Source

```
<wml>
  <card>
  <p> hello </p>
    <do type=”accept”> <go href =”wtai://wp/mc;+33147466667”/> </do>
    <p> world </p>
  </card>
</wml>
```

Result :

The text “hello world” is displayed (the user may have to press “OK” between “hello” and “world”). When the user presses on “accept”, the call is set up.

5.1.2 SetupCall

5.1.2.1 TEST_TELEPHONY_WTAI_SetupCall.01

Description

Test WTAI fucntion “setup call” from WTAI voice control library.

Source

```
<wml>
  <card>
        <do type=”accept”>
          <go href =”wtai://vc/sc;+33111111111”/>
        </do>
    <p> hello world </p>
  </card>
</wml>
```

Result

If the deck comes from a trusted HTTP server, the same as for TEST_TELEPHONY_WTAI_MakeCall.01. Otherwise, no SBC is generated and a deck explaining the error is sent to the browser.

5.1.3 SendDTMF

5.1.3.1 TEST_TELEPHONY_WTAI_PublicSendDtmf.01

Description

Test WTAI fucntion “setup call” from WTAI voice control library.

Source

```
<wml>
```

```
    <card>
        <p> hello</p>
                <do type=”accept” name=”setupcall”>
                    <go href =”wtai://wp/mc;331…”/>
                </do>
                <do type=”accept” name=”sendDTMF”>
                    <go href =”wtai://wp/sd;12”/>
                </do>
        <p> world </p>
    </card>
</wml>
```

Result :

The text “hello world” is displayed. Then the user presses on the “setupcall” softkey. The call is set up. The user presses on the “sendDTMF” softkey. The DTMF is sent.

5.1.3.2 TEST_TELEPHONY_WTAI_VCsendDtmf.01

Description

It is the “send dtmf” function of the “voice call control” library.

Source

```
<wml>
    <card>
        <p> hello world </p>
                <do type=”accept” name=”setupCall”>
                    <go href =”wtai://wp/mc;+33147466667”/>
                </do>
                <do type=”accept” name=”sendDTMF”>
                    <go href =”wtai://vc/sd;12”/>
                </do>
    </card>
</wml>
```

Result :

If the deck comes from a untrusted HTTP server, the gateway sends a deck explaining that an error occurred.

Otherwise, the result is the same as for TEST_TELEPHONY_WTAI_PublicSendDtmf.01.

5.1.4 Send USSD

5.1.4.1 TEST_TELEPHONY_WTAI_GSMSendUssd.01

Description

Test the “send ussd” function of the “gsm” library.

Source

```
<wml>
   <card>
      <p> hello </p>
            <do type=”accept”>
               <go href =”wtai://gsm/su;*#157#;;;”/>
            </do>
            <p> world </p>
   </card>
</wml>
```

Result :

If the deck comes from an untrusted HTTP server, the gateway sends a special deck to the browser, indicating an error occurred. Otherwise, the text “hello world” is displayed, and, when the user presses on the “accept” button, the USSD is sent.

6 S@TML STK extensions

In this chapter, we assume that the HTTP server hosting S@TML decks is a trusted server.

6.1 Variables and constants

6.1.1 Constants

6.1.1.1 TEST_CONSTANT.01

Source :

```
<satml>
   <sat-const
   sat-name=”welcome”
   sat-value=”hello”/>
   <card>
      <p> $(sat-const:welcome) world !</p>
   </card>
```

```
</satml>
```

Result :

The text “hello world” is displayed.

6.1.1.2 TEST_CONSTANTS.02

Description

Aim : test to set the value of a constant.

Assign a run time value to a constant

Source :

```
<satml>
    <sat-const
    sat-name = ”welcome”
    sat-value = ”hello”/>
    <card>
        <p> hello
        <input name=”sat-const:welcome“/></p>
    </card>
</satml>
```

Result

The gateway sends an “error deck” to the gateway.

6.1.1.3 TEST_CONSTANTS.03

Description

Naming of variables and constants with an empty variable

Source

```
<satml>
    <sat-const
    sat-name = ”welcome”
    sat-value = ”hello”/>
    <card>
        <p> $(welcome) world </p>
    </card>
</satml>
```

Result

The text "world" is displayed.

6.1.1.4 TEST_CONSTANTS.04

Description

Naming of variables and constants with a variable having a specified value

Source :

```
<satml>
   <sat-const
   sat-name = ”welcome”
   sat-value = ”world”/>
   <card>
      <p>
      enter the text string you want
      <input name=”welcome”/>
      $(welcome) world</p>
   </card>
</satml>
```

Result

- The text "enter the text string you want" is displayed.
- The user enters "squirrel"
- The text "squirrel world" is displayed.

6.1.2 SPS

6.1.2.1 TEST_SPS.01

Description

Test SPS (optional feature).

Source

```
<satml
   sat-serv-id=”03”>
   <sat-sps
   sat-name=”x”
   sat-var-id=”02”/>
   <card>
      <p> $(sat-sps:x) world !</p>
```

```
    </card>

</satml>
```

Result :

There are two possibilities, according to the capabilities of the gateway to support sps or not :

- If the gateway does not support sps, an error deck is sent to the browser, explaining the error.
- If the gateway supports sps, the SBC is generated (the tests for sps, if supported by the gateway and the browser, are out of the scope of this document).

6.1.3 Variables

6.1.3.1 Test_temporaryVariable_clearing.01

Description

Context : the two decks described below are stored on the same HTTP server. The name of the first can be anything, but the second deck must be called "second.wml".

Source

FIRST DECK

```
<satml>

    <sat-var

        sat-name="x"

        sat-do-clear="true"/>

    <card>

        <p>

            enter the value for x <input name="x"/> <br/>

            <a href="second.wml"> go to second deck </a>

        </p>

    </card>

</satml>
```

SECOND DECK (on the same HTTP server as the first one. Its name is second.wml).

```
<satml>

    <card>

        <p> the value of x is $(x). </p>

    </card>

</satml>
```

Result

The text "enter the value of x" is displayed. He enters "sim". Then the user selects the URL "go to second deck". The text "the value of x is" is displayed.

6.2 Fields

6.2.1 <sat-play-tone>

6.2.1.1 TEST_EXTENSIONS_sat-play-tone.01

Description

Normal context, minimum parameter set

Source

```
<satml>
  <card>
    <p>
      <sat-play-tone sat-tone=”beep”/>
    </p>
  </card>
</satml>
```

Result

ME shall play tone with manufacturer default duration.

6.2.1.2 TEST_EXTENSIONS_sat-play-tone.02

Description

Normal context, title attribute present

Source

```
<satml>
  <card>
    <p>
      <sat-play-tone sat-title=”playing tone...” sat-tone=”beep” />
    </p>
  </card>
</satml>
```

Result

ME shall play tone with manufacturer default duration and the string “playing tone ...” must be displayed.

6.2.1.3 TEST_EXTENSIONS_sat-play-tone.03

Description

Normal context, full parameter set

Source

```
<satml>

  <card>

    <p>

      <sat-play-tone sat-title=”playing tone...” sat-duration=”100”
      sat-tone=”beep” />

    </p>

  </card>

</satml>
```

Result

ME shall play tone with duration of 10 seconds and the string “playing tone ...” must be displayed.

6.2.2 <sat-inkey>

6.2.2.1 TEST_EXTENSIONS_INKEY.01

Description

Simple inkey field with no parameters.

Source

```
<satml>
  <card id="tink01">
    <p>
      inkey: <sat-inkey sat-name="inkey"/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying “*inkey:*” and then shall read one symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard. It shall then display the given input e.g. “*inkey value is: A*” (or “*f*” *or* “*9*” *or* “*é*”).

6.2.2.2 TEST_EXTENSIONS_INKEY.02

Description

Inkey field with format parameters.

Source

```
<satml>
  <card id="tink02" >
    <p>
      inkey: <sat-inkey sat-name="inkey" sat-format="1M"/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*inkey:*" and then shall read one symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard. It shall then display the given input e.g. "*inkey value is: A*" (or "*f*" *or* "*9*" *or* "*é*").

6.2.2.3 TEST_EXTENSIONS_INKEY.03

Description

Inkey field with format parameters.

Source

```
<satml>
  <card id="tink03" >
    <p>
      inkey: <sat-inkey sat-name="inkey" sat-format="M"/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

As in test case 6.2.2.2 SB shall prompt for an input by displaying "*inkey:*" and then shall read one symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard. It shall then display the given input e.g. "*inkey value is: A*" (or "*f*" *or* "*9*" *or* "*é*").

6.2.2.4 TEST_EXTENSIONS_INKEY.04

Description

Inkey field with format parameters.

Source

```
<satml>
  <card id="tink04" >
    <p>
      inkey: <sat-inkey sat-name="inkey" sat-format="1Y"/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*inkey:*" and then shall allow to make a positive or negative decision It shall then display the given input e.g. "*inkey value is: 01*" for a positive or "*00*" for a negative decision. (If the handset does not support the GSM 11.14 $(BinChoice)$ feature an error is displayed.)

6.2.2.5 TEST_EXTENSIONS_INKEY.05

Description

Inkey field with format parameters.

Source

```
<satml>
  <card id="tink05" >
    <p>
      inkey: <sat-inkey sat-name="inkey" sat-format="Y"/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

As in test case 6.2.2.4 SB shall prompt for an input by displaying "*inkey:*" and then shall allow to make a positive or negative decision It shall then display the given input e.g. "*inkey value is: 01*" for a positive or "*00*" for a negative decision. (If the handset does not support the GSM 11.14 $(BinChoice)$ feature an error is displayed.)

6.2.2.6 TEST_EXTENSIONS_INKEY.06

Description

Inkey field with format parameters.

Source

```
<satml>
  <card id="tink06" >
    <p>
      inkey: <sat-inkey sat-name="inkey" sat-format="N"/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*inkey:*" and then shall read one numeric characters (including "+", "*", "#") at the keyboard. It shall then display the given input e.g. "*inkey value is: 4*" or "+".

6.2.2.7 TEST_EXTENSIONS_INKEY.07

Description

Inkey field with format parameters.

Source

```
<satml>
  <card id="tink07" >
    <p>
     inkey: <sat-inkey sat-name="inkey" sat-format="1N"/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

As in test case 6.2.2.6 SB shall prompt for an input by displaying "*inkey:*" and then shall read one numeric characters (including "+", "*", "#") at the keyboard. It shall then display the given input e.g. "*inkey value is: 4*" or "+".

6.2.2.8 TEST_EXTENSIONS_INKEY.8

Description

Inkey field with bad format parameters.

Source

```
<satml>
  <card id="tink08" >
    <p>
      inkey: <sat-inkey sat-name="inkey" sat-format="*M"/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

SB should display an error detected by the DE as this format is not defined for the inkey field.

6.2.2.9 TEST_EXTENSIONS_INKEY.9

Description

Simple inkey field with title parameter.

Source

```
<satml>
  <card id="tink09">
    <p>
      <sat-inkey sat-name="inkey" sat-title="inkey:"/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

As in test case 6.2.2.1.

6.2.2.10 TEST_EXTENSIONS_INKEY.10

Description

Simple inkey field with title parameter defined by variable.

Source

```
<satml>
  <card id="tink10">
    <p>
      <input name="mytit" title="title:"/>
      <sat-inkey sat-name="inkey" sat-title="$mytit:"/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*title:*" and then shall read any symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard. It shall then prompt again for an input by displaying the entered string, e.g. *"Inkey:"*. After reading one symbolic, numeric, uppercase or lowercase alphabetic character at the keyboard it shall then display the given input e.g. "*inkey value is: 6*" or "x".

6.2.2.11 TEST_EXTENSIONS_INKEY.11

Description

Inkey field with format parameters and help string.

Source

```
<satml>
  <card id="tink11" >
    <p>
      inkey: <sat-inkey sat-name="inkey" sat-format="1N"
                    sat-help="Enter one digit."/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*inkey:*" and then shall accept one numeric characters (including "+", "*", "#"). When the user presses the help button while asked for input it shall display the text "Enter one digit" assuming that the contextual menu is in default state. After leaving the help display the prompt for input shall be re-displayed. When one character has been entered the SB shall display the given input e.g. "*inkey value is: 8*".

6.2.2.12 TEST_EXTENSIONS_INKEY.12

Description

Inkey field with format parameters and help strings.

Source

```
<satml sat-help="Look into the Browser manual.">
  <card id="tink12"
        sat-help="Fill in the inkey fields.">
    <p>
      inkey: <sat-inkey sat-name="inkey" sat-format="1N"
                    sat-help="Enter one digit."/>
      inkey value is: $inkey
    </p>
  </card>
</satml>
```

Result

As in test case 6.2.2.11.

6.3 Statements

6.3.1 <sat-send-sms>

6.3.1.1 TEST_EXTENSIONS_sat-send-sms.01

Description

Normal context, minimum parameter set

Source

```
<satml>

  <card>
```

```
    <sat-send-sms sat-dest=”+49170123456”>
        SMS Test
    </sat-send-sms>
  </card>
</satml>
```

Result

ME shall send SMS “SMS Test” to destination `”+49170123456”>`.

6.3.1.2 TEST_EXTENSIONS_sat-send-sms.02

Description

Normal context, full parameter set.

Source

```
<satml>
  <card>
    <sat-send-sms sat-title=”sending sms...”
        sat-smsc="+4912345787”
        sat-dest=”+3312545666”>
          SMS Test
    </sat-send-sms>
  </card>
</satml>
```

Result

ME shall send SMS “SMS Test” to destination `”+3312545666” using the SMSC "+4912345787”`. While this message is being sent the string “sending sms...” must be displayed.

6.3.1.3 TEST_EXTENSIONS_sat-send-sms.03

Description

Normal context, minimum parameter set.

Error case: Text > 160 characters

Source

```
<satml>
  <card>
    <sat-send-sms sat-dest=”+4912345678”>
        12345678901234567890123456789012345678901234567890
        12345678901234567890123456789012345678901234567890
        12345678901234567890123456789012345678901234567890
        12345678901234567890123456789012345678901234567890
    </sat-send-sms>
  </card>
</satml>
```

Result

Browser shall truncate SMS text and send 140 characters to destination address.

6.3.1.4 TEST_TELEPHONY_SML_SendSMS.04

Description

Source

```
<satml>
  <card>
    <p> hello </p>
    <sat-send-sms sat-dest="+33147466667"> how are you ? </sat-send-sms>
    <p> world </p>
  </card>
</satml>
```

Result :

The text “hello world” is displayed, and after a single user interaction, the SMS “how are you” is sent to the phone number indicated, using the SMSC number of the ME. Then “world” is displayed.

6.3.2 <sat-refresh>

6.3.2.1 TEST_EXTENSIONS_sat-refresh.01

Description

SIM Reset

Source

```
<satml>
  <card>
    <p>SIM reset</p>
    <sat-refresh sat-cmdqual=”04” /> <!-- SIM reset -->
  </card>
</satml>
```

Result

SIM must be reset.

6.3.3 <sat-exit>

6.3.3.1 TEST_EXTENSIONS_sat-exit.01

Description

Exit the browser

Source

```
<satml>
  <card>
    <p>Exit browser</p>
    <sat-exit/> <!-- exit browser -->
  </card>
</satml>
```

Result

The browser session must be canceled.

6.3.4 <sat-gen-stk>

6.3.4.1 TEST_EXTENSIONS_sat-gen-stk.01

Description

Coding of the STK DISPLAY TEXT command using the `sat-`gen-stk element. This commands sends a diaplay text with normal priority, with clear done by user. The text to display is “hello world”.

Source

```
<satml>
  <card>
    <sat-gen-stk
    sat-cmdtype=”21”
    sat-cmdqual=”80”
    sat-destdev=”02”
    sat-data=”8D 0C 04 48 65 6C 6C 6F 20 57 6F 72 6C 64”/>
  </card>
</satml>
```

Result

The string “Hello World” must be displayed.

6.3.4.2 TEST_EXTENSIONS_sat-gen-stk.02

Description

Coding of the STK DISPLAY TEXT command using the `sat-`gen-stk element. The command type given is not a valid type.

Source

```
<satml>
  <card>
    <sat-gen-stk
    sat-cmdtype=”FF”
    sat-cmdqual=”80”
    sat-destdev=”02”
    sat-data=”8D 0C 04 48 65 6C 6C 6F 20 57 6F 72 6C 64”/>
  </card>
</satml>
```

Result

ME should issue error message.

6.3.4.3 TEST_EXTENSIONS_sat-gen-stk.03

Description

Coding of the STK DISPLAY TEXT command using the `sat-`gen-stk element. The command sent is a “display text” with text “hello world”. But the destination device is not the correct one.

Source

```
<satml>
  <card>
    <sat-gen-stk
    sat-cmdtype=”21”
    sat-cmdqual=”80”
    sat-destdev=”FF”
    sat-data=”8D 0C 04 48 65 6C 6C 6F 20 57 6F 72 6C 64”/>
  </card>
</satml>
```

Result

ME should issue error message.

6.3.4.4 TEST_EXTENSIONS_sat-gen-stk.04

Description

Coding of the STK DISPLAY TEXT command using the `sat-`gen-stk element. The command sent is a “display text” with text “hello world”. But the “L” indicated in sat-data is not correct.

Source

```
<satml>
  <card>
    <sat-gen-stk
    sat-cmdtype=”21”
    sat-cmdqual=”80”
    sat-destdev=”02”
    sat-data=”8D 02 04 48 65 6C 6C 6F 20 57 6F 72 6C 64”/>
  </card>
</satml>
```

Result

Browser must issue error message.

6.3.4.5 TEST_EXTENSIONS_sat-gen-stk.05

Description

Coding of the STK DISPLAY TEXT command using the `sat-`gen-stk element. The command sent is a “display text” with text “hello world”. But the “L” indicated in sat-data is not correct.

Source

```
<satml>
  <card>
    <sat-gen-stk
    sat-cmdtype=”21”
    sat-cmdqual=”80”
    sat-destdev=”02”
    sat-data=”8D 7F 04 48 65 6C 6C 6F 20 57 6F 72 6C 64”/>
  </card>
</satml>
```

Result

Browser must issue error message.

6.3.5 <sat-setup-call>

In all test for this part, the MSISDN indicated must be changed to one which is managed by the person runing the tests.

6.3.5.1 TEST_TELEPHONY_SML_SetupCall.01

Description

Test <sat-setup-call> command with default attributes.

Source

<wml>

```
    <card>
        <p> hello </p>
        <sat-setup-call  sat-dest=”+33147466667”/>
        <p> world </p>
    </card>
</wml>
```

Result

The text “hello” is displayed. Then the user is asked for confirmation (with a default string) :

- if the user accepts the call, the call is set up and “world” is displayed.

- if the user does not accept the call, “world” is displayed.

6.3.5.2 TEST_TELEPHONY_SML_SetupCall.02

Description

Test <sat-setup-call> with attributes set in the SATML source.

Source

```
<satml>
    <card>
        <p> hello </p>
        <sat-setup-call
        sat-confirm=”please confirm you want to set up the call”
        sat-dest=”+33147466667”
        sat-title=”call has been set up”/>
        <p> world </p>
    </card>
</satml>
```

Result :

We assume we are working with a handset supporting the two alpha identifiers “user confirmation phase” and “call set up phase”.

The text “hello” is displayed and “please confirm you want to set up a call” is displayed.

- if the user accepts the call, it is set up and “world” is displayed.
- If the user rejects the call, “world” is displayed.

6.3.5.3 TEST_TELEPHONY_SML_SetupCall.03

Description

Test <sat-setup-call> with command qualifier changed for the second call.

Source

```
<satml>
    <card>
        <p> hello </p>
        <sat-setup-call
        sat-dest=”+33147465648” />
        <sat-setup-call
        sat-cmdqual=”02”
        sat-dest=”+33147466667” />
        <p> world </p>
    </card>
</satml>
```

Result :

The text “hello” is displayed and the user is asked for confirmation by a default text. The user accepts the first call. He is asked for confirmation for the second call.

- if the user accepts this call, then the first call is put on hold and “world” is displayed.
- If the user rejects this call, “world” is displayed.

6.3.5.4 TEST_TELEPHONY_SML_SetupCall.04

Description

Test <sat-setup-call> with DTMF sent at the same time.

Source

```
<satml>
    <card>
        <p> hello </p>
        <sat-setup-call
        sat-dest=”+33147465648,1234” />
        <p> world </p>
    </card>
</satml>
```

Result :

The text “hello” is displayed and the user is asked for confirmation by a default text. The user accepts to set up the call. The call is sent, with “1234” as DTMF.

6.3.6 <sat-send-ussd>

6.3.6.1 TEST_TELEPHONY_SML_SendUSSD.01

Description

Test the tag <sat-send-ussd>

Source

```
<satml>
   <card>
      <p> hello </p>
      <sat-send-ussd
      sat-data=*#157#/>
      <p> world </p>
   </card>
</satml>
```

Result :

If the deck comes from an untrusted HTTP server, the gateway sends a special deck to the browser, indicating an error occurred. Otherwise, the text “hello world” is displayed, and, at the same time, the USSD is sent.

6.3.7 <sat-local-info>

6.3.7.1 TEST_TELEPHONY_SML_LocalInfo.01

Description

Test tag <sat-local-info>.

Source

```
<satml>
   <card>
      <p> hello </p>
      <sat-local-info
      sat-name=”x”/>
      <p> the value of x is $x</p>
   </card>
</satml>
```

Result

The text “hello” is displayed. After a user interaction, “the value of x is <binary string>” is displayed, where <binary string> contains the binary string for the STK terminal response “provide local info”.

6.3.8 <sat-encrypt>

6.3.8.1 TEST_ENCRYPTION.01

Description

Authentication of user input by DES CBC cryptographic checksum.

Source

```
<satml>
  <card id="tenc01">
    <!-- Browser calculates MAC -->
    <input name=”amount” title=”amount (in EUR):” format=”*N”/>
    <sat-encrypt sat-check="mac"
                 sat-kid="01"
                 sat-inlist="BUY BNR=12345678 AMT=,$(amount), EUR"
                 sat-out="sec_msg"/>
    Press OK to accept.
    <do type="accept">
      <go method="post" href="http://www.squirrel.net/buy.satml">
        <postfield name="msg" value="$sec_msg"/>
      </go>
    </do>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying “*amount (in EUR):*” and then shall read any digit characters (including ‘+’, ‘*’, ‘#’) at the keyboard until the (user agent dependant) default maximum length is reached, e.g. “42”. Then, for the LV-formatted strings “BUY BNR=12345678 AMT=”, “42”, “ EUR” a cryptographic checksum will be calculated using the DES CBC algorithm, 00 for padding, and the key field with index 0. A secure message will be constructed with MAC element and clear text data block (see /SBC 1.0.x/). The text “*press OK to accept”* is displayed. If the user invokes the accept function a short message is sent to the DE, where it will be forwarded to the named URL using the post method of HTTP.

6.3.8.2 TEST_ENCRYPTION.02

Description

Encryption of user input using DES CBC mode.

Source

```
<satml>
  <card id="tenc02">
    <!-- Browser encrypts -->
    <input name=”amount” title=”amount (in EUR):” format=”*N”>
    <sat-encrypt sat-crypt=”true”
                 sat-kic=”01”
                 sat-inlist="BUY BNR=12345678 AMT=,$(amount), EUR"
                 sat-out="sec_msg"/>
    Press OK to accept.
    <do type="accept">
      <go method="post" href="http://www.squirrel.net/buy.satml">
         <postfield name="msg" value="$sec_msg"/>
      </go>
    </do>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*amount (in EUR):*" and then shall read any digit characters (including '+', '*', '#') at the keyboard until the (user agent dependant) default maximum length is reached, e.g. "42". Then, the string "SIGN BNR=12345678 AMT=42 EUR" will be encrypted using the DES CBC algorithm, 00 for padding, and the key field with index 0. A secure message will be constructed with enciphered data block, but without MAC element (see /SBC 1.0.x/). The text "*press OK to accept"* is displayed. If the user invokes the accept function a short message is sent to the DE, where it will be forwarded to the named URL using the post method of HTTP.

6.3.8.3 TEST_ENCRYPTION.03

Description

Encryption of user input using DES ECB mode.

Source

```
<satml>
  <card id="tenc03">
    <!-- Browser encrypts -->
    <input name=”amount” title=”amount (in EUR):” format=”*N”>
    <sat-encrypt sat-crypt=”true”
                 sat-kic=”0D”
                 sat-inlist="BUY BNR=12345678 AMT=,$(amount), EUR"
                 sat-out="sec_msg"/>
    Press OK to accept.
    <do type="accept">
      <go method="post" href="http://www.squirrel.net/buy.satml">
         <postfield name="msg" value="$sec_msg"/>
      </go>
    </do>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*amount (in EUR):*" and then shall read any digit characters (including '+', '*', '#') at the keyboard until the (user agent dependant) default maximum length is reached, e.g. "42". Then, the string "SIGN BNR=12345678 AMT=42 EUR" is encrypted using the DES ECB algorithm, 00 for padding, and the key field with index 0. A secure message will be constructed with enciphered data block, but without MAC element (see /SBC 1.0.x/). The text "*press OK to accept"* is displayed. If the user invokes the accept function a short message is sent to the DE, where it will be forwarded to the named URL using the post method of HTTP.

6.3.8.4 TEST_ENCRYPTION.04

Description

Authentication by cryptographic checksum and encryption using DES.

Source

```
<satml>
  <card id="tenc04">
    <!-- Browser calculates MAC and encrypts -->
    <input name=”amount” title=”amount (in EUR):” format=”*N”>
    <sat-encrypt sat-check="mac"
```

```
                sat-kid="01"
                sat-crypt="true"
                sat-kic="11"
                sat-inlist="BUY BNR=12345678 AMT=,$(amount), EUR"
                sat-out="sec_msg"/>
    Press OK to accept.
    <do type="accept">
      <go method="post" href="http://www.squirrel.net/buy.satml">
         <postfield name="msg" value="$sec_msg"/>
      </go>
    </do>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*amount (in EUR):*" and then shall read any digit characters (including '+', '*', '#') at the keyboard until the (user agent dependant) default maximum length is reached, e.g. "42". Then, for the string "SIGN BNR=12345678 AMT=42 EUR" a cryptographic checksum will be calculated using the DES CBC algorithm, 00 for padding, and the key field with index 0. The data will be encrypted using DES in CBC mode, 00 for padding, and with the key field with index 1. A secure message will be constructed with MAC element and enciphered data block (see /SBC 1.0.x/). The text "*press OK to accept"* is displayed. If the user invokes the accept function a short message is sent to the DE, where it will be forwarded to the named URL using the post method of HTTP.

6.3.8.5 TEST_ENCRYPTION.05

Description

Authentication by cryptographic checksum and encryption using Triple DES, outer CBC, 2 keys, for both KIC and KID.

Source

```
<satml>
  <card id="tenc05">
    <!-- Browser calculates MAC and encrypts -->
    <input name="amount" title="amount (in EUR):" format="*N">
    <sat-encrypt sat-check="mac"
                sat-kid="05"
                sat-crypt="true"
                sat-kic="15"
                sat-inlist="BUY BNR=12345678 AMT=,$(amount), EUR"
                sat-out="sec_msg"/>
    Press OK to accept.
    <do type="accept">
      <go method="post" href="http://www.squirrel.net/buy.satml">
         <postfield name="msg" value="$sec_msg"/>
      </go>
    </do>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying "*amount (in EUR):*" and then shall read any digit characters (including '+', '*', '#') at the keyboard until the (user agent dependant) default maximum length is reached, e.g. "42". Then, for the string "SIGN BNR=12345678 AMT=42 EUR" a cryptographic checksum will be calculated using the Triple DES outer-CBC mode algorithm with 2 keys, 00 for padding, and the key field with index 0. The data will be encrypted using Triple DES in outer-CBC mode with 2 keys, 00 for padding, and with the key field with index 1. A secure message will be constructed with MAC element and enciphered data block (see /SBC 1.0.x/). The text "*press OK to accept"* is displayed. If the user invokes the accept function a short message is sent to the DE, where it will be forwarded to the named URL using the post method of HTTP.

6.3.8.6 TEST_ENCRYPTION.06

Description

Secure message format without security.

Source

```
<satml>
  <card id="tenc06">
    <!-- Browser produces secure message format, only -->
    <input name=”amount” title=”amount (in EUR):” format=”*N”>
    <sat-encrypt sat-inlist="BUY BNR=12345678 AMT=,$(amount), EUR"
                 sat-out="insec_msg"/>
    Press OK to accept.
    <do type="accept">
      <go method="post" href="http://www.squirrel.net/buy.satml">
        <postfield name="msg" value="$insec_msg"/>
      </go>
    </do>
  </card>
</satml>
```

Result

SB shall prompt for an input by displaying “*amount (in EUR):*” and then shall read any digit characters (including ‘+’, ‘*’, ‘#’) at the keyboard until the (user agent dependant) default maximum length is reached, e.g. “42”. Then, for the string “SIGN BNR=12345678 AMT=42 EUR” a secure message will be constructed without MAC element and clear text data block (see /SBC 1.0.x/). The text “*press OK to accept”* is displayed. If the user invokes the accept function a short message is sent to the DE, where it will be forwarded to the named URL using the post method of HTTP.

6.3.9 <sat-decrypt>

6.3.9.1 TEST_DECRYPTION.01

Description

Authentication of server data by DES CBC cryptographic checksum.

Source

```
<satml>
  <card id="tdec01">
    <!-- Browser verifies MAC -->
    <sat-decrypt sat-check=”mac”
                 sat-kid=”01”
                 sat-mac=”A4 12 4D 44 9D 36 B0 83”
                 sat-in=”1E
                    15 42 55 59 20 42 4E 52 3D 31 32 33 34 35 36 37 38
                    37 38 20 41 4D 54 3D 02 34 32 04 20 45 55 52”
                 sat-outlist=”txt,val,curr”/>
                 <!-- KID=0x01: key 0, DES CBC -->
    <p>The secured message is:<br/>$txt $val $curr</p>
  </card>
</satml>
```

Result

SB gets the given constant string as a secure message with MAC element and clear text data block (see /SBC 1.0.x/). A cryptographic checksum will be calculated using the DES CBC algorithm, 00 for padding, and the key field with index 0. If the authentication fails the SB will stop. Else (for KID key 0101010101010101) the text "*The secured message is:*" " BUY BNR=12345678 AMT=42 EUR" will be displayed.

6.3.9.2 TEST_DECRYPTION.02

Description

Decryption of server data using DES CBC mode.

Source

```
<satml>
  <card id="tdec02">
    <!-- Browser decrypts -->
    <sat-decrypt sat-crypt=”true”
                 sat-kic=”01”
                 sat-in=”20
                    9C F3 2F 9B 97 A1 B6 12 72 4E 70 C8 7F 88 AE 27
                    5B C4 BD C0 C9 75 7A 5A A4 12 4D 44 9D 36 B0 83”
                 sat-outlist="txt,val,curr"/>
                 <!-- KIc=0x01: key 0, DES CBC -->
    The secured message is:<br>$txt $val $curr
  </card>
</satml>
```

Result

SB gets the given constant string as a secure message with enciphered data block, but without MAC element (see /SBC 1.0.x/). It will be decrypted using the DES CBC algorithm, and the key field with index 0. After decryption (with KIc key 0101010101010101) and removing the padding, the text "*The secured message is:*" " BUY BNR=12345678 AMT=42 EUR" will be displayed.

6.3.9.3 TEST_DECRYPTION.03

Description

Decryption of server data using DES ECB mode.

Source

```
<satml>
  <card id="tdec03">
    <!-- Browser decrypts -->
    <sat-decrypt sat-crypt=”true”
                 sat-kic=”0D”
                 sat-in=”20
                    9C F3 2F 9B 97 A1 B6 12 A0 9B 77 B3 7D 7B FC 08
                    A7 E1 AF 31 DB D0 D0 5C DC 7A FB E3 09 C0 36 75”
                 sat-outlist="txt,val,curr"/>
                 <!-- KIc=0x0D: key 0, DES ECB -->
    The secured message is:<br>$txt $val $curr
  </card>
</satml>
```

Result

SB gets the given constant string as a secure message with enciphered data block, but without MAC element. (see /SBC 1.0.x/). It will be decrypted using the DES ECB algorithm, and the key field with index 0. After decryption and removing the padding the text "*The secured message is:" "* BUY BNR=12345678 AMT=42 EUR*"* will be displayed.

6.3.9.4 TEST_DECRYPTION.04

Description

Decryption of server data and authentication using DES.

Source

```
<satml>
  <card id="tdec04">
    <!-- Browser decrypts and verifys MAC -->
    <sat-decrypt sat-check="mac"
                 sat-kid="01"
                 sat-crypt=”true”
                 sat-kic=”11”
                 sat-mac=”69 C0 9D 85 9C 27 7A 5A”
                 sat-in="20
                    AA CC AB 62 5B D0 E7 ED 15 2B AB D2 93 E1 4D E1
                    FF FA 70 E9 8B 4F 20 C8 B8 59 75 80 D6 6B EC 13”
                 sat-outlist="txt,val,curr"/>
                 <!-- KID=0x01: key 0, DES CBC -->
                 <!-- KIc=0x11: key 1, DES CBC -->
    The secured message is:<br>$txt $val $curr
  </card>
</satml>
```

Result

SB gets the given constant string as a secure message with MAC element and enciphered data block (see /SBC 1.0.x/). It will be decrypted using the DES CBC algorithm, and the key field with index 0. A cryptographic checksum will be calculated using the DES CBC algorithm, 00 for padding, and the key field with index 1. If the authentication fails the SB will stop. Else (for KID and KIc key 0101010101010101) the text "*The secured message is:" "* BUY BNR=12345678 AMT=42 EUR*"* will be displayed.

6.3.9.5 TEST_DECRYPTION.05

Description

Decryption of server data and authentication using Triple DES.

Source

```
<satml>
  <card id="tdec05">
    <!-- Browser decrypts and verifys -->
    <sat-decrypt sat-check="mac"
                 sat-kid="05"
                 sat-crypt=”true”
                 sat-kic=”15”
                 sat-mac=”00 25 B2 8E C3 67 79 E8”
```

```
                sat-in="20
                   B5 BF BF 03 54 BA 33 1A 7E C7 76 42 E3 59 6F 55
                   B8 CF 9A 33 83 7C D5 82 FE E1 3A D1 67 9D FF 28”
                sat-outlist="txt,val,curr"/>
                <!-- KID=0x05: key 0, 3DES, outer-CBC, 2 keys -->
                <!-- KIc=0x15: key 1, 3DES, outer-CBC, 2 keys -->
    The secured message is:<br>$txt $val $curr
  </card>
</satml>
```

Result

SB gets the given constant string as a secure message with MAC element and enciphered data block (see /SBC 1.0.x/). It will be decrypted using the Triple DES outer-CBC mode algorithm with two keys, and the key field with index 0. A cryptographic checksum will be calculated using the Triple DES outer-CBC mode algorithm, 00 for padding, and the key field with index 1. If the authentication fails the SB will stop. Else (for KID and KIc key 0101010101010101-1010101010101010) the text “*The secured message is:” “* BUY BNR=12345678 AMT=42 EUR*”* will be displayed.

6.3.9.6 TEST_DECRYPTION.06

Description

Secure message format without security.

Source

```
<satml>
  <card id="tdec06">
    <!-- Browser gets secure message format, only -->
    <sat-decrypt sat-in="&#24;&#7;amount=
                        &#5;12.34
                        &#3;EUR
                        &#0;&#0;&#0;&#0;&#0;&#0”
                sat-outlist="txt,val,curr"/>
    The secured message is:<br>$txt $val $curr
  </card>
</satml>
```

Result

SB gets the given constant string as a secure message with clear text data block and without MAC element (see /SBC 1.0.x/). It will be returned in the given variables, and the text “*The secured message is:” “amount= 12.34 EUR”* will be displayed.

6.3.10 <sat-plug-in>

6.3.10.1 TEST_EXTENSIONS_sat-plug-in.01

Description

Use the ConvertTextPhoneNumberToGSMPhoneNumber plug in.

Source

```
<satml>
  <card id="tplug01">
    enter the phone number:
    <input name=”numstr” format=”*N”/>
    <sat-plug-in sat-uid=”FF01” sat-inlist=”$(numstr)” sat-outlist=”numgsm”/>
    <sat-setup-call sat-title=”calling $(numstr)” sat-dest=”$(numgsm)”/>
  </card>
</satml>
```

Result

The SB must prompt for input with the string *“enter the phone number:”* and allow the user to enter a string of digits (including "+", "*", "#"). Then the number is displayed, e.g. *"calling +443344556677"* and a call set-up is started. However, according to SATML V1.0.4 it is also possible that SB displays an error message produced by DE as it is currently stated there that no variables can be used in the `sat-dest` attribute of `sat-setup-call`.

6.3.10.2 TEST_EXTENSIONS_sat-plug-in.02

Description

Use an invalid plug-in command

Error case: Invalid sat-uid.

Source

```
<satml>
  <card id="tplug02">
    enter the phone number:
    <input name=”numstr” format=”*N”/>
    <sat-plug-in sat-uid=”FF77” sat-inlist=”$(numstr)” sat-outlist=”numgsm”/>
    <sat-setup-call sat-title=”calling $(numstr)” sat-dest=”$(numgsm)”/>
  </card>
</satml>
```

Result

SB should issue error message.

6.3.10.3 TEST_EXTENSIONS_sat-plug-in.03

Description

Use an invalid plug-in command

Error case: Invalid sat-inlist.

Source

```
<satml>
  <card id="tplug03">
    enter the phone number:
    <input name=”numstr” format=”*N”/>
    <sat-plug-in sat-uid=”FF01” sat-inlist=”$(numstr),$(numstr)”
      sat-outlist=”numgsm”/>
    <sat-setup-call sat-title=”calling $(numstr)” sat-dest=”$(numgsm)”/>
  </card>
</satml>
```

Result

SB should issue error message.

6.3.10.4 TEST_EXTENSIONS_sat-plug-in.04

Description

Use an invalid plug-in command

Error case: Invalid sat-outlist.

Source

```
<satml>
  <card id="tplug04">
    enter the phone number:
    <input name=”numstr” format=”*N”/>
    <sat-plug-in sat-uid=”FF01” sat-inlist=”$(numstr)”
      sat-outlist=”numgsm,tralala”/>
    <sat-setup-call sat-title=”calling $(numstr)” sat-dest=”$(numgsm)”/>
  </card>
</satml>
```

Result

SB should issue error message.

7 Annex A : future tests

7.1 Variables use

7.1.1.1 TEST_TELEPHONY_SML_SendSMS_future.01

Description

Source

```
<satml>
  <card>
    <p> hello </p>
    <!--the validity period of the SMS is set to be 10 days, 12 hours and 13 minutes -->
    <sat-send-sms
      sat-title=”send an SMS”
      sat-dest=”+33147466667”
      sat-period=”0101213”>
       how are you ?
      </sat-send-sms>
    <p> world </p>
  </card>
</satml>
```

Result :

The text “hello” is displayed and after a single user interaction, “send an SMS” is displayed (if this is supported by the handset). Then, “world” is displayed. The SMS “how are you” as been sent to the phone number indicated in sat-dest. In the “send sms” command generated by the gateway, the period must is coded “176”, which stands for “10 days” (the number of hours and minutes is ignored).

7.1.1.2 TEST_EXTENSIONS_sat-send-sms_future.02

Description

Normal context, full parameter set

Source

```
<satml>
  <card>
  <p>
    <input title=”SMSC Number?” name=”smscnr” format=”*N”/>

    <input title=”Destination Number?” name=”destnr” format=”*N”/>
  </p>
    <sat-send-sms sat-title=”sending sms...”
         sat-smsc=”$smscnr”
         sat-dest=”$destnr”>
           SMS Test
    </sat-send-sms>
  </card>
</satml>
```

Result

ME shall ask user with the string “SMSC Number?” for the number of the short message service center and with “Destination Number?” for the destination address of the short message and shall send SMS “SMS Test” to this destination. While this message is being sent the string “sending sms...” must be displayed.