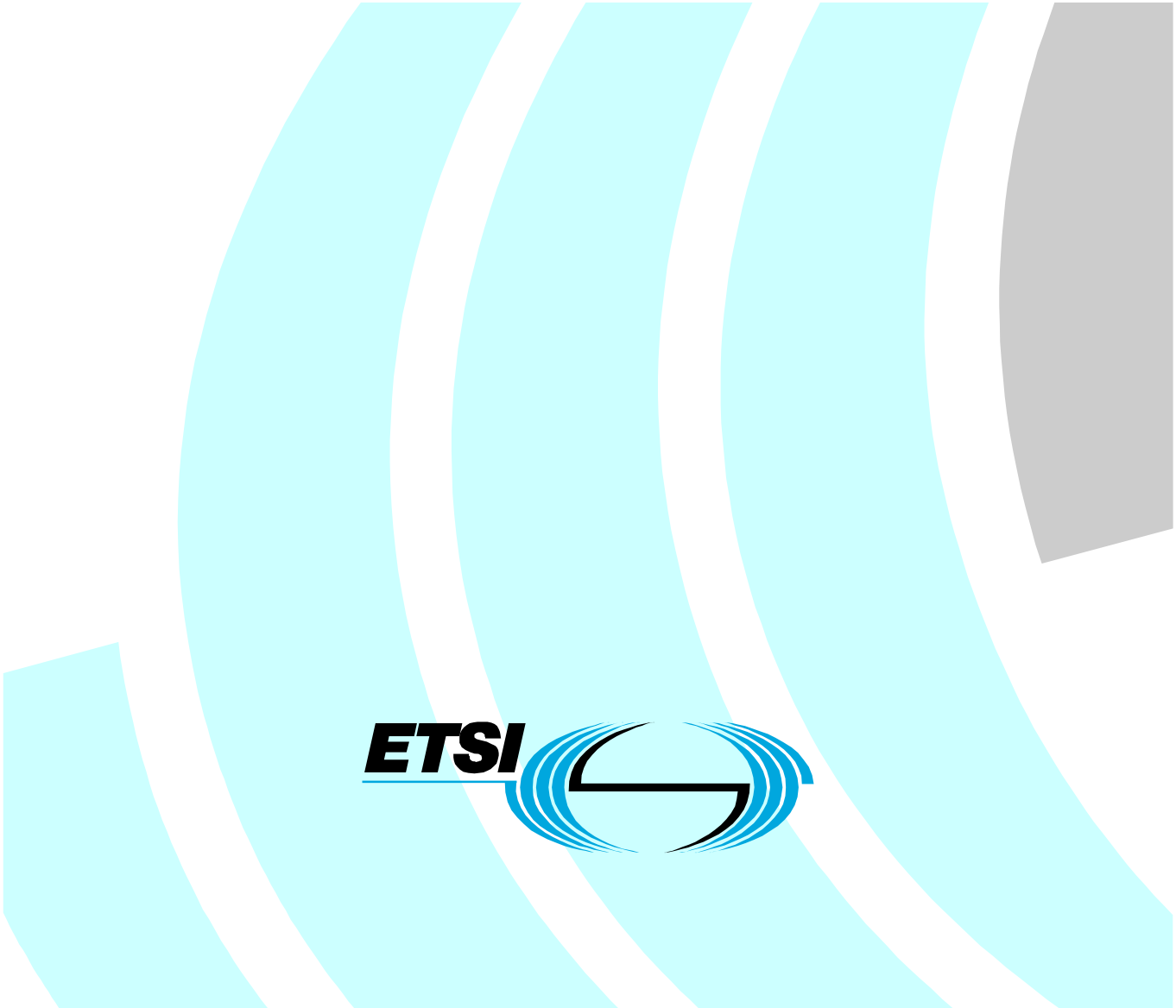
ETSI TR 102 171 V1.1.1 (2003-02)

Technical Report

Using ECMA-323 (CSTA XML) in a Voice Browser Environment



Reference
DTR/ECMA-00285

Keywords
CSTA, voice

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

All published ETSI deliverables shall include information which directs the reader to the above source of information.

Foreword

This Technical Report (TR) has been produced by ECMA on behalf of its members and those of the European Telecommunications Standards Institute (ETSI).

Brief History

The present document illustrates how CSTA XML (ECMA-323) can be used in a Voice Browser environment. The present document is part of a suite of ECMA CSTA Phase III Standards and Technical Reports.

All of the Standards and Technical Reports in this Suite are based upon the practical experience of ECMA member companies and each one represents a pragmatic and widely based consensus.

1 Scope

Services for Computer Supported Telecommunications Applications are defined by Standard ECMA-269 and the XML Protocol for those services is defined by Standard ECMA-323.

In many cases, applications require only a small subset of the features standardized in CSTA. In a voice browser environment, processing speech (not call control) is usually the major focus of the application. For example, from a CSTA feature perspective, an application may simply need to answer an incoming call and then later clear it. As these speech-centric applications evolve they can use additional, more advanced, features standardized by CSTA that are provided by CSTA-conformant communications platforms.

Since ECMA-269 and ECMA-323 are relatively large standards (combined over 1100 pages), it is a challenge for application developers without prior knowledge of the CSTA standards to know where to find basic concepts that they need to understand in order to implement basic CSTA features.

The present document illustrates how ECMA-323 can be used in a Voice Browser environment. These concepts illustrated in the present document can be applied to any Voice Browser environment that provides an XML-based read/write messaging interface (i.e. CSTA Service Boundary) that supports asynchronous events from a CSTA conformant communication platform. SALT enabled browsers that implement a ECMA-323 interface for call control using the SALT smex mechanism is an example of a browser with this capability.

Throughout the present document the term "ECMA-323 enabled voice browser" is used, in a generic sense, to refer to browser implementation that support a CSTA conformant ECMA-323 interface.

Examples are provided that show how ECMA-323 can be used in several different environments such as SALT-enabled browsers and CCXML.

2 References

The present document provides informative examples of how to use ECMA-323 in a Voice Browser environment. The following ECMA Standards should be used as the definitive references for CSTA.

ECMA-269: "Services for Computer Supported Telecommunications Applications (CSTA) Phase III".

ECMA-323: "XMLProtocol for Computer Supported Telecommunications Applications (CSTA) Phase III".

ECMA CSTA Standards can be used for call control in many different environments. The following references provide additional information on using the ECMA CSTA standards in different environments:

SALT: "Speech Application Language Tags Specification Version 1.0", SALT Forum, 15 July 2002. (http://www.saltforum.org).

CCXML: "Voice Browser Call Control: CCXML Version 1.0", W3C Working Draft, 11 October 2002. (http://www.w3c.org/TR/ccxml/).

3 Brief Overview of ECMA-323

ECMA-323 consists of a set of XML Schemas based upon the W3C XML Schema Language Recommendation. The Standard includes schemas for many categories of services defined in ECMA-269.

Call control is just one category of services in ECMA-323. Examples of other categories of services are: capability exchange (feature discovery) services, call routing services, services to control a device (e.g. message waiting, writing to display, forwarding settings), and many others.

CSTA provides a protocol independent abstraction layer for applications. It provides a consistent, standards-based messaging interface that can be used with basic 1st party call control based platforms as well as more complex 3rd party call control (CTI) platforms, or a combination of both (1st party call control with some additional 3rd party call control features).

CSTA modelling and concepts are also compatible with many procedural and object models such as the SALT CallControl object (chapter 3 of the SALT specification).

4 Fundamental Concepts

This clause introduces some informative modelling concepts that are useful to illustrate how ECMA-323 enabled voice browsers can use ECMA-323 messages. The actual ECMA CSTA standards should be used for the definitive descriptions.

4.1 CSTA Connection

CSTA call control services are applied to CSTA connections. A CSTA connection refers to a relationship between a call and a telephony endpoint. A CSTA connection is referenced via a CSTA connection identifier. A CSTA connection identifier consists of a call identifier and a device (endpoint) identifier.

In a typical 1st party call control implementation, a voice browser application manipulates only the CSTA connection directly associated with the voice browser platform. However, other call control implementations may also provide application control of other endpoints in the call using CSTA services (via 3rd party call control, for example). A device identifier is included in a CSTA connection identifier to allow any endpoint to be addressed by a voice browser application.

4.2 CSTA Connection State Model

A ECMA-323 enabled voice browser application is informed of connection state transitions (via ECMA-323 call control events) by placing a monitor on a telephony endpoint via an associated address (e.g. this is how an application "listens" for incoming calls).

Each CSTA connection in a call is associated with a connection state. CSTA specifies a connection state model (see ECMA-269, figure 6-19) that consists of the following connection states:

- Alerting: Indicates an incoming call at an endpoint. Typically the connection may be ringing or it may be in a pre-alerting (e.g. offered) condition.
- Connected: Indicates that a connection is actively participating in a call. This connection state can be the result of an incoming or outgoing call.
- Failed: Indicates that call progression has stalled. Typically this could represent that an outgoing call attempt that encountered a busy endpoint.
- Held: Indicates that an endpoint is no longer actively participating in a call. For implementations that support multiple calls per endpoint (i.e. line), a connection could be Held while the line is used to place another call (consultation transfer on an analogue line, for example).
- Initiated: A transient state, usually indicating that the endpoint is initiating a service (e.g. dialtone).
- Null: There is no relationship between the call and the endpoint.
- Queued: Indicates that the call is temporarily suspended at a device (e.g. call has been parked, camped on).

The CSTA Connection State is provided in ECMA-323 events.

4.3 Connection State Transitions for CSTA Calls

4.3.1 Incoming Call

The following figure illustrates the CSTA events for an incoming call. The connection state of endpoint on the voice browser platform (called connection) is indicated in parenthesis.

- Delivered Event (Alerting): Indicates call is alerting. Calls that are "auto-answered" do not sent this event. A CSTA Answer Call service can be used to answer the call. This results in an Established event.
- Established Event (Connected): Indicates call has been answered. Media path has been established. The CSTA Clear Connection service can be used to clear the call. A Connection Cleared event is generated as the result of the Clear Connection service.
- Connection Cleared Event (Null): Indicates connection has cleared. This can be the result of the Clear Connection service or as the result of any party clearing from the call.

4.3.2 Outgoing Call

The following figure illustrates the CSTA events for an outgoing call. The connection state of the endpoint on the Voice Browser platform (originating connection) is indicated in parenthesis. This sequence could be the result of a CSTA Make Call service.

- Originated Event (Connected): Indicates that the originating connection (an endpoint on the voice browser platform) is connected.
- Delivered Event (Connected): Indicates the call is alerting the called party.
- Established Event (Connected): Indicates the called party has answered the call. Media path has been established.
- Connection Cleared Event (Null): Indicates connection has cleared.

5 CSTA Profiles

Since many CSTA features are optional, and to enhance application portability across different CSTA implementations, CSTA standards require a minimal subset of functionality as conformance criteria.

ECMA-269 specifies a set of Profiles. At least one profile is required to be supported. The following profiles most closely match the call control services and events needed by a Voice Browser application.

- Level 1a Voice Browser Profile (added in ECMA-269 5th edition): Provides support for answering an incoming call, clearing, and moving the call to another endpoint using the Single Step Transfer Call service. The Get Switching Function Capabilities Service is not required to be supported in this profile.
- Level 1b Voice Browser Profile (added in ECMA-269 5th edition): Provides support for answering an incoming call, clearing, and moving the call to another endpoint using the Deflect Call service. The Get Switching Function Capabilities Service is not required to be supported in this profile.
- Level 2 Voice Browser Profile (added in ECMA-269 5th edition): Provides support for making a call in addition to the services and events required in either the Level1a Voice Browser Profile of the Level 1b Voice Browser Profile. The Get Switching Function Capabilities Service is required to be supported in this profile.
- Basic Telephony Profile: Provides support for answering an incoming call, creating an outgoing call, and clearing the call. The Get Switching Function Capabilities Service is required to be supported in this profile.

NOTE: Telephony platforms that interface with networks and/or endpoints that do not expose the underlying network/device signalling are not expected to provide all of these CSTA events. For example, if the telephony network does not provide a busy indication, the Failed event is not required.

5.1 Level 1a Voice Browser Profile

5.1.1 Services

The following CSTA services are included in the *Level 1a Voice Browser Profile*:

- Answer call: Answers an alerting call. In a voice browser, the answering device is an endpoint on the platform.
- Clear Connection: Clears a connection. In a voice browser environment, the clearing device is an endpoint on the voice browser platform.
- Single Step Transfer (of a connected call): Transfers a call to another endpoint. In a voice browser environment, the transferring device is an endpoint on the voice browser platform and is no longer involved with the call after the single step transfer service is completed.
- Monitor Start: Establishes a device-type monitor on an endpoint. In a voice browser environment, the monitored device is an endpoint on the voice browser platform.
- Monitor Stop: Terminates an existing monitor.

5.1.2 Events

The following CSTA events are included in the *Level 1a Voice Browser Profile*:

- Connection Cleared: Indicates that an endpoint has disconnected from a call.
- Delivered: Indicates that a call is alerting an endpoint.
- Established: Indicates that an endpoint has answered or been connected to a call.
- Failed: Indicates that a call cannot be completed (e.g. call has encountered a busy endpoint).
- Transferred: Indicates that an existing call has been transferred from an endpoint (on the voice browser platform) to another endpoint and has been disconnected from the call. This implies that the transferring device connection state is Null: no Connection Cleared event is generated for the transferring device after the Transferred event.

5.2 Level 1b Voice Browser Profile

5.2.1 Services

The following CSTA services are included in the *Level 1b Voice Browser Profile*:

- Answer call: Answers an alerting call. In a voice browser, the answering device is an endpoint on the platform.
- Clear Connection: Clears a connection. In a voice browser environment, the clearing device is an endpoint on the voice browser platform.
- Deflect (of a connected call): Moves a connection away from the deflecting device. In a voice browser environment, the deflecting device is an endpoint on the voice browser platform and is no longer involved with the call after the Deflect Call service is completed.
- Monitor Start: Establishes a device-type monitor on an endpoint. In a voice browser environment, the monitored device is an endpoint on the voice browser platform.
- Monitor Stop: Terminates an existing monitor.

5.2.2 Events

The following CSTA events are included in the *Level 1b Voice Browser Profile*:

- Connection Cleared: Indicates that an endpoint has disconnected from a call.
- Delivered: Indicates that a call is alerting an endpoint.
- Diverted: Indicates that the endpoint (on the voice browser platform) has redirected a call to another endpoint and is no longer involved with the call.
- Established: Indicates that an endpoint has answered or been connected to a call.
- Failed: Indicates that a call cannot be completed (e.g. call has encountered a busy endpoint).

5.3 Level 2 Voice Browser Profile

5.3.1 Services

In this profile, a CSTA implementation is required to provide its capabilities to applications via the ECMA-323 Get Switching Function Capabilities service. The capabilities include the list of ECMA-323 services and events supported by a telephony platform and the various types of behaviour options supported by an implementation (the profile(s) supported, the types of digits that are allowed in the dialling string for an outbound call, etc.). Many of the parameters are optional and do not need to be provided in the ECMA-323 message.

The following CSTA services are included in the *Level 2 Voice Browser Profile*:

- Answer call: Answers an alerting call. In a voice browser, the answering device is an endpoint on the voice browser platform.
- Clear Connection: Clears a connection. In a voice browser environment, the clearing device is an endpoint on the voice browser platform.
- Make Call: Establishes a call between two devices. In a voice browser environment, the originating device is an endpoint on the voice browser platform.
- Monitor Start: Establishes a device-type monitor on an endpoint. In a voice browser environment, the monitored device is an endpoint on the voice browser platform.
- Monitor Stop: Terminates an existing monitor.

In addition, at least one of the following services must be supported in order to move a connected call away from an endpoint:

- Single Step Transfer (of a connected call): Transfers a call to another endpoint. In a voice browser environment, the transferring device is an endpoint on the voice browser platform and is no longer involved with the call after the single step transfer service is completed.
- Deflect (of a connected call): Moves a connection away from the deflecting device. In a voice browser environment, the deflecting device is an endpoint on the voice browser platform and is no longer involved with the call after the Deflect Call service is completed.

5.3.2 Events

The following CSTA events are included in the *Level 2 Voice Browser Profile:*

- Connection Cleared: Indicates that an endpoint has disconnected from a call.
- Delivered: Indicates that a call is alerting an endpoint.
- Established: Indicates that an endpoint has answered or been connected to a call.

- Failed: Indicates that a call cannot be completed (e.g. call has encountered a busy endpoint).
- Network Reached: For an outbound call, indicates that the call has been connected to an external network via a Network Interface Device.
- Originated: For an outbound call, indicates that the originating endpoint (on the voice browser platform) is connected to the call.

In addition:

- Diverted (if the Deflect Service was used to move a connected call).
- Transferred (if the Single Step Transfer Service was used to move a connected call)

5.4 Basic Telephony Profile

5.4.1 Services

In this profile, the CSTA implementation is required to provide its capabilities to applications via the ECMA-323 Get Switching Function Capabilities service. The capabilities include the list of ECMA-323 services and events supported by a telephony platform and the various types of behaviour options supported by an implementation (the profile(s) supported, the types of digits that are allowed in the dialling string for an outbound call, etc.). Many of the parameters are optional and do not need to be provided in the ECMA-323 message.

The following CSTA services are included in the *Basic Telephony Profile*:

- Answer call: Answers an alerting call. In a voice browser, the answering device is an endpoint on the platform.
- Clear Connection: Clears a connection. In a voice browser environment, the clearing device is an endpoint on the voice browser platform.
- Make Call: Establishes a call between two devices. In a voice browser environment, the originating device is an endpoint on the voice browser platform.
- Monitor Start: Establishes a monitor on an endpoint. In a voice browser environment, the monitored device is an endpoint on the voice browser platform.
- Monitor Stop: Terminates an existing monitor.

5.4.2 Events

The following CSTA events are included in the *Basic Telephony Profile:*

- Connection Cleared: Indicates that an endpoint has disconnected from a call.
- Delivered: Indicates that a call is alerting an endpoint.
- Established: Indicates that an endpoint has answered or been connected to a call.
- Failed: Indicates that a call cannot be completed (e.g. call has encountered a busy endpoint).
- Network Reached: For an outbound call, indicates that the call has been connected to an external network via a Network Interface Device.
- Originated: For an outbound call, indicates that the originating endpoint (on the voice browser platform) is connected to the call.
- Service Initiated: Indicates that the endpoint (on the voice browser platform) is requesting service (i.e. dialtone). Some endpoints bypass this state and are not expected to provide this event (e.g. endpoints that do not provide dialtone).

5.5 Other Features

The services defined in these profiles are a very small subset of the call control features standardized in CSTA. A telephony platform may provide any number of additional ECMA-323 services and may even provide features that have not been standardized in ECMA-323 by using the ECMA-323 extension mechanism.

The following ECMA-323 services and events may also be very useful for voice browser applications:

- Single Step Conference service: This service adds another device into an existing call. Unlike the Single Step Transfer service, no devices are dropped from the call as a result of this service.
- Conferenced event: Indicates that a device has been added to an existing call and that no devices have been removed from the existing call.
- Get Switching Function Devices service: This service provides the list of endpoint identifiers that are provided by the telephony platform. If this service is not supported, a voice browser application will have to determine the endpoint identifiers outside of the ECMA-323 interface.

6 ECMA-323 Illustrative Examples

This clause provides some examples of ECMA-323 XML messages (W3C Instance Documents) in the context of voice browser usage scenarios. Examples include:

- discovering the features supported by a telephony platform;
- establishing a monitor on a device (i.e. listen for incoming calls);
- notification of an inbound alerting connection;
- answering an inbound connection, notification of a connected connection;
- clearing a connection, notification of a cleared connection;
- initiating an outbound call;
- a single step transfer scenario;
- a single step conference scenario.

6.1 Discovering the Capabilities of a Telephony Platform

The ECMA-323 Get Switching Function Capabilities service can be used by a voice browser application to determine the features (services and events) that a telephony platform supports.

See ECMA-323, clause 11.1.3, for the format of the information provided.

6.2 Starting a Monitor (i.e. listening for incoming calls)

In order to receive ECMA-323 call control events, a CSTA monitor must be placed on a telephony endpoint via the ECMA-323 Monitor Start service.

6.2.1 Monitor Start - Service Request example

This example shows how a voice browser application places a monitor on an endpoint that is associated with the identifier 22343. Note that by omitting the ECMA-323 optional parameters, the most relevant type of monitoring (a device-type monitor on a device) is requested.

The mandatory Service Request parameters are:

- monitorObject: the device identifier of the endpoint. This can either be obtained by ECMA-323 (see 6.1) or by mechanisms outside of ECMA-323.

```
<?xml version="1.0" encoding="UTF-8"?>
<MonitorStart xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <monitorObject>
      <deviceObject>22343</deviceObject>
 </monitorObject>
</MonitorStart>
```

6.2.2 Monitor Start - Service Response example

This example shows the response to the Monitor Start request. The monitorCrossRefID of 99 will be provided in all ECMA-323 events associated with this monitor request.

The mandatory Service Response parameters are:

- monitorCrossRefIdentifier: this parameter provides a handle to correlate subsequent ECMA-323 events with this monitor request. Since each ECMA-323 event contains this handle, it can be used to associate the event to the voice browser application that issued the Monitor Start request.

```
<?xml version="1.0" encoding="UTF-8"?>
<MonitorStartResponse xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <monitorCrossRefID>99</monitorCrossRefID>
</MonitorStartResponse>
```

6.3 Notification of an Inbound Call

When an inbound call arrives at a monitored endpoint, an ECMA-323 Delivered event is provided by the telephony platform.

The Delivered event contains information about the call such as ANI and DNIS information provided by the network for example. This information enables a voice browser application to determine if it should answer the call or reject it, how the call should be handled, etc.

6.3.1 Delivered Event example

The event contains the following parameters:

- monitorCrossRefID: This parameter represents the handle provided in the Monitor Start response.
- connection: This parameter contains connection identifier that is used in CSTA services that are to be applied to this connection (e.g. Answer Call).
- alertingDevice: This parameter provides the endpoint identifier of the alerting connection. For inbound calls, it is the endpoint on the voice browser platform. For outbound calls, the alerting device represents the called device that is alerting.
- callingDevice: This is the local representation of the calling device. Due to features in the telephony platform (transfers, forwarding, conferences, etc.) the calling device may have changed before it has reached the voice browser platform.
- calledDevice: This is the local representation of the called device. Due to features in the telephony platform (transfers, forwarding, conferences, etc.) the called device may have changed before it has reached the voice browser platform.
- lastRedirectionDevice: Indicates if the call has been previously redirected.
- localConnectionState: Indicates the connection state of "alerting" at the near end connection. This will typically be the connection associated with the monitored device on the voice browser platform.

- cause: Indicates the cause of the event. In this case, it represents a "new" call.
- networkCallingDevice: For inbound calls, this represents the calling device information provided by the network (ANI, for example). This information will always remain with the call and will never change (unlike the callingDevice).
- networkCalledDevice: For inbound calls, this represents the called device information provided by the network (DNIS, for example). This information will always remain with the call and will never change (unlike the calledDevice).
- associatedCallingDevice: This parameter is provided when the call in an inbound call. The network interface device identifier is 023 in this example. This could also be provided with the value of "notKnown".

```
<?xml version="1.0" encoding="UTF-8"?>
<DeliveredEvent xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <monitorCrossRefID>99</monitorCrossRefID>
 <connection>
      <callID>1</callID>
      <deviceID>22343</deviceID>
 </connection>
 <alertingDevice>
      <deviceIdentifier>22343</deviceIdentifier>
 </alertingDevice>
 <callingDevice>
      <deviceIdentifier>14085551212</deviceIdentifier>
 </callingDevice>
 <calledDevice>
      <deviceIdentifier>22343</deviceIdentifier>
 </calledDevice>
 <lastRedirectionDevice>
      <notRequired/>
 </lastRedirectionDevice>
 <localConnectionInfo>alerting</localConnectionInfo>
 <cause>newCall</cause>
 <networkCallingDevice>
      <deviceIdentifier>14085551212</deviceIdentifier>
 </networkCallingDevice>
 <networkCalledDevice>
      <deviceIdentifier>18001234567</deviceIdentifier>
 </networkCalledDevice>
 <associatedCallingDevice>
      <deviceIdentifier>023</deviceIdentifier>
 </associatedCallingDevice>
</DeliveredEvent>
```

6.4 Answering an Inbound Call

After the voice browser application is notified of the incoming call it uses the ECMA-323 Answer Call service to answer the call.

6.4.1 Answer Call - Service Request example

The mandatory Service Request parameters are:

- callToBeAnswered: The connection identifier of the alerting connection. The application uses the contents of the connection parameter provided in the Delivered event.

```
<?xml version="1.0" encoding="UTF-8"?>
<AnswerCall xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <callToBeAnswered>
      <callID>1</callID>
      <deviceID>22343</deviceID>
 </callToBeAnswered>
</AnswerCall>
```

6.4.2 Answer Call - Service Response example

The telephony platform provides a positive response to the Answer Call service request. There are no mandatory Service Request parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<AnswerCallResponse xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2"/>
```

6.5 Notification of a Connected Call

As the result of the Answer Call service, an ECMA-323 Established event is provided by the telephony platform to indicate that the connection is now in the connected state.

If the telephony platform was configured to "auto-answer" the call, the Established event would be the first event sent to the voice browser application.

6.5.1 Established Event example

Since the Established event contains similar information as in Delivered event example, the parameter description sections is omitted.

```
<?xml version="1.0" encoding="UTF-8"?>
<EstablishedEvent xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <monitorCrossRefID>99</monitorCrossRefID>
 <establishedConnection>
      <callID>1</callID>
      <deviceID>22343</deviceID>
 </establishedConnection>
 <answeringDevice>
      <deviceIdentifier>22343</deviceIdentifier>
 </answeringDevice>
 <callingDevice>
      <deviceIdentifier>14085551212</deviceIdentifier>
 </callingDevice>
 <calledDevice>
      <deviceIdentifier>22343</deviceIdentifier>
 </calledDevice>
 <lastRedirectionDevice>
      <numberDialed/>
 </lastRedirectionDevice>
<localConnectionInfo>connected</localConnectionInfo>
 <cause>normal</cause>
 <networkCallingDevice>
      <deviceIdentifier>14085551212</deviceIdentifier>
 </networkCallingDevice>
 <networkCalledDevice>
      <deviceIdentifier>18001234567</deviceIdentifier>
 </networkCalledDevice>
 <associatedCallingDevice>
      <deviceIdentifier>023</deviceIdentifier>
 </associatedCallingDevice>
</EstablishedEvent>
```

6.6 Clearing a Connection

Once the voice browser application is finished with the call, it sends an ECMA-323 Clear Connection service to the telephony platform.

6.6.1 Clear Connection - Service Request example

The mandatory Service Request parameters are:

- connectionToBeCleared: the connection identifier of the connection to be cleared. The application uses the connection identifier provided by the telephony platform in previous events.

```
<?xml version="1.0" encoding="UTF-8"?>
<ClearConnection xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <connectionToBeCleared>
      <callID>1</callID>
      <deviceID>22343</deviceID>
 </connectionToBeCleared>
</ClearConnection>
```

6.6.2 Clear Connection - Service Response example

The telephony platform provides a positive response to the Clear Connection service request. There are no mandatory Service Request parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<ClearConnectionResponse xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2"/>
```

6.7 Notification of a Cleared Connection

As the result of the Clear Connection service, an ECMA-323 Connection Cleared event is provided by the telephony platform to indicate that the connection is now cleared.

6.7.1 Connection Cleared Event example

The event contains the following parameters:

- monitorCrossRefID: this parameter represents the handle provided in the Monitor Start response.
- droppedConnection: this parameter contains connection identifier of the connection that has been cleared.
- releasingDevice: this parameter provides the endpoint identifier of the releasing device.
- localConnectionState: indicates the connection state of "null" at the near end connection. This will typically be the connection associated with the monitored device on the voice browser platform.
- cause: indicates the cause of the event. In this case, it represents a "normal" clearing.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConnectionClearedEvent xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <monitorCrossRefID>99</monitorCrossRefID>
 <droppedConnection>
      <callID>1</callID>
      <deviceID>22343</deviceID>
 </droppedConnection>
 <releasingDevice>
      <deviceIdentifier>22343</deviceIdentifier>
 </releasingDevice>
 <localConnectionInfo>null</localConnectionInfo>
 <cause>normalClearing</cause>
</ConnectionClearedEvent>
```

Note that the voice browser application should also be prepared to receive a Connection Cleared event that indicates that the far end endpoint has released the call (i.e. far end disconnects first).

6.8 Initiating an Outbound Call

A voice browser application can initiate an outbound call by using the ECMA-323 Make Call Service. Before it initiates an outbound call, the application should place a monitor on the originating device, in order to monitor the progress of the call.

6.8.1 Make Call - Service Request example (refer to Profile)

In this example, the application wants to establish a call from the device identifier 22343 (on the voice browser platform) to an external device associated with the telephone number 18001234567.

The mandatory Service Request parameters are:

- callingDevice: the device identifier associated with the calling device on the voice browser platform (22343).
- calledDevice: represents the endpoint associated with the telephone number 18001234567.
- autoOriginate: By setting the value to "doNotPrompt" it indicates that the calling device connection should automatically be answered. This mode should always be used for devices associated with the voice browser platform.

```
<?xml version="1.0" encoding="UTF-8"?>
<MakeCall xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <callingDevice>22343</callingDevice>
 <calledDirectoryNumber>18005551212</calledDirectoryNumber>
 <autoOriginate>doNotPrompt</autoOriginate>
</MakeCall>
```

6.8.2 Make Call - Service Response example

This example shows the response to the Make Call request.

The mandatory Service Response parameters are:

- callingDevice connection identifier: this parameter provides the connection identifier that is used to reference the connection in subsequent services. It is also provided in future events associated with the connection.

```
<?xml version="1.0" encoding="UTF-8"?>
<MakeCallResponse xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <callingDevice>
      <callID>2</callID>
      <deviceID>22343</deviceID>
 </callingDevice>
</MakeCallResponse>
```

6.9 Outbound Call Event Sequence

This example shows the sequence of events generated as the result of the Make Call service request.

The event sequence includes the following events: Originated, Network Reached, Delivered, and Established.

6.9.1 Originated Event Example

The Originated event indicates that the calling device (associated with the voice browser platform) is connected to the call.

The event contains the following parameters:

- monitorCrossRefID: this parameter represents the handle provided in the Monitor Start response.
- originatedConnection: this parameter contains connection identifier of the originating connection on the voice browser platform. This is used in CSTA services that are to be applied to this connection.

- callingDevice: this is the local representation of the calling device associated with the voice browser platform.
- calledDevice: this is the representation of the called device, in this example, the phone number 18005551212.
- localConnectionState: indicates the connection state of "connected" at the originated connection. Note that this is not the connection state of the called device's connection.

```
<?xml version="1.0" encoding="UTF-8"?>
<OriginatedEvent xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <monitorCrossRefID>99</monitorCrossRefID>
 <originatedConnection>
      <callID>2</callID>
      <deviceID>22343</deviceID>
 </originatedConnection>
 <callingDevice>
      <deviceIdentifier>22343</deviceIdentifier>
 </callingDevice>
 <calledDevice>
      <deviceIdentifier>18005551212</deviceIdentifier>
 </calledDevice>
 <localConnectionInfo>connected</localConnectionInfo>
 <cause>makeCall</cause>
</OriginatedEvent>
```

6.9.2 Network Reached Event

The Network Reached event indicates that the call has encountered a network interface device (CO line, trunk, etc.).

The event contains the following parameters:

- monitorCrossRefID: this parameter represents the handle provided in the Monitor Start response.
- outboundConnection: this parameter contains connection identifier of the outbound connection associated with the Network interface device.
- networkInterfaceUsed: indicates the endpoint identifier of the network interface device.
- callingDevice: this is the local representation of the calling device associated with the voice browser platform.
- calledDevice: this is the representation of the called device, in this example, the phone number 18005551212.
- lastRedirectionDevice: "notKnown" is provided to indicate that there is no known redirection number.
- localConnectionState: indicates the connection state of "connected" at the near end connection (typically the originatedConnection). Note that this is not the connection state of the called device's connection.
- cause: cause of "normal" is provided

```
<?xml version="1.0" encoding="UTF-8"?>
<NetworkReachedEvent xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <monitorCrossRefID>99</monitorCrossRefID>
 <outboundConnection>
      <callID>2</callID>
      <deviceID>023</deviceID>
 </outboundConnection>
 <networkInterfaceUsed>
      <deviceIdentifier>023</deviceIdentifier>
 </networkInterfaceUsed>
 <callingDevice>
      <deviceIdentifier>22343</deviceIdentifier>
 </callingDevice>
 <calledDevice>
      <deviceIdentifier>18005551212</deviceIdentifier>
 </calledDevice>
 <lastRedirectionDevice>
      <notRequired/>
```

```
 </lastRedirectionDevice>
 <localConnectionInfo>connected</localConnectionInfo>
 <cause>normal</cause>
</NetworkReachedEvent>
```

6.9.3 Delivered Event

If the underlying signalling can detect that the called device is alerting, the telephony platform provides a Delivered event.

The event contains the following parameters:

- monitorCrossRefID: this parameter represents the handle provided in the Monitor Start response.
- connection: this parameter contains connection identifier that is associated with the far end device that is alerting. For outbound calls, this is the connection to the network interface device, which represents the called device.
- alertingDevice: this parameter provides the endpoint identifier of the alerting connection. For outbound calls, the alerting device represents the called device that is alerting.
- callingDevice: this is the local representation of the calling device (device associated with the voice browser platform).
- calledDevice: this is the called Device.
- lastRedirectionDevice: indicates if the call has been previously redirected. In this example "NotRequired" is provided.
- localConnectionState: indicates the connection state of "connected" for the connection on the voice browser platform. Note that this is not the connection state of the called device's connection.
- cause: indicates the cause of the event. In this case, it represents a "networkSignal" which indicates that the event represents the far end device as opposed to the network interface device.
- associatedCalledDevice: this parameter is provided when the call in an outbound call. The network interface device identifier is 023 in this example. This could also be provided with the value of "notKnown".

```
<?xml version="1.0" encoding="UTF-8"?>
<DeliveredEvent xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <monitorCrossRefID>99</monitorCrossRefID>
 <connection>
      <callID>2</callID>
      <deviceID>023</deviceID>
 </connection>
 <alertingDevice>
      <deviceIdentifier>18005551212</deviceIdentifier>
 </alertingDevice>
 <callingDevice>
      <deviceIdentifier>22343</deviceIdentifier>
 </callingDevice>
 <calledDevice>
      <deviceIdentifier>18005551212</deviceIdentifier>
 </calledDevice>
 <lastRedirectionDevice>
      <notRequired/>
 </lastRedirectionDevice>
 <localConnectionInfo>connected</localConnectionInfo>
 <cause>networkSignal</cause>
 <associatedCalledDevice>
      <deviceIdentifier>023</deviceIdentifier>
 </associatedCalledDevice>
</DeliveredEvent>
```

6.9.4 Established Event

The Established event is provided by the telephony platform when it detects that the called party has answered the call. The ability to detect that the call device has answered is dependent upon the underlying signalling capabilities.

Since the Established event is very similar to the Delivered event, it is not shown in this example.

6.10 Single Step Transfer

A voice browser application can use the ECMA-323 Single Step Transfer service to transfer the caller away from the voice browser platform to another endpoint. In the process of transferring the call, the transferring connection at the voice browser platform is disconnected from the call.

6.10.1 Single Step Transfer - Service Request example

The mandatory Service Request parameters are:

- activeCall connection: the connection identifier of the transferring connection (connection at the voice browser platform).
- transferredTo: the device identifier associated with the transferred to endpoint.

```
<?xml version="1.0" encoding="UTF-8"?>
<SingleStepTransferCall xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <activeCall>
      <callID>1</callID>
      <deviceID>22343</deviceID>
 </activeCall>
 <transferredTo>333333</transferredTo>
</SingleStepTransferCall>
```

6.10.2 Single Step Transfer - Service Response example

The telephony platform provides a positive response to the Single Step Transfer service request.

The mandatory Service Response parameters are:

- transferredCall connection: the connection identifier at the transferredTo connection.

```
<?xml version="1.0" encoding="UTF-8"?>
<SingleStepTransferCallResponse xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <transferredCall>
      <callID>2</callID>
      <deviceID>333333</deviceID>
 </transferredCall>
</SingleStepTransferCallResponse>
```

6.11 Notification of a Transferred Connection

As the result of the Single Step Transfer service, an ECMA-323 Transferred event is provided by the telephony platform to indicate that the transferring connection is now cleared and the transferred device is now connected to the transferred-to device.

6.11.1 Transferred Event example

The event contains the following parameters:

- monitorCrossRefID: this parameter represents the handle provided in the Monitor Start response.
- primaryOldCall connection: this parameter contains connection identifier of the connection of the transferring endpoint in the voice browser platform.

- transferringDevice: this parameter provides the endpoint identifier of the transferring device on the voice browser platform.
- transferredToDevice: this parameter represents the transfer-to device (i.e. the new device in the call).
- transferredConnections: for single step transfers, this includes the old connection at the transferring device that has been cleared as the result of the transfer.
- localConnectionState: indicates the connection state of "null" at the local connection on the voice browser platform.
- cause: indicates the cause of the event. In this case, it indicates that the transfer is a single step transfer.

```
<?xml version="1.0" encoding="UTF-8"?>
<TransferedEvent xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <monitorCrossRefID>99</monitorCrossRefID>
 <primaryOldCall>
      <callID>1</callID>
      <deviceID>22343</deviceID>
 </primaryOldCall>
 <transferringDevice>
      <deviceIdentifier>22343</deviceIdentifier>
 </transferringDevice>
 <transferredToDevice>
      <deviceIdentifier>333333</deviceIdentifier>
 </transferredToDevice>
 <transferredConnections>
      <connectionListItem>
           <oldConnection>
                <callID>1</callID>
                <deviceID>22343</deviceID>
           </oldConnection>
      </connectionListItem>
 </transferredConnections>
 <localConnectionInfo>null</localConnectionInfo>
 <cause>singleStepTransfer</cause>
</TransferedEvent>
```

Note that the voice browser application will not see a Connection Cleared event for the Transferring device after a Transferred event: the Transferred event indicates the transferring device connection has already cleared from the call.

6.12 Deflect

A voice browser application can use the ECMA-323 Deflect service to move a call away from the voice browser platform to another endpoint.

6.12.1 Deflect - Service Request example

The mandatory Service Request parameters are:

- callToBeDiverted connection: the connection identifier that is to be moved (connection at the voice browser platform).
- newDestination: the device identifier associated with the new destination.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeflectCall xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <callToBeDiverted>
      <callID>1</callID>
      <deviceID>22343</deviceID>
 </ callToBeDiverted >
 <newDestination>333333</ newDestination>
</DeflectCall>
```

6.12.2 Deflect - Service Response example

The telephony platform provides a positive response to the Deflect service request. There are no mandatory Service Request parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeflectCallResponse xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2"/>
```

6.13 Notification of a Diverted Connection

As the result of the Deflect service, an ECMA-323 Diverted event is provided by the telephony platform to indicate that the deflected connection is now cleared and the call has been moved to another device.

6.13.1 Diverted Event example

The event contains the following parameters:

- monitorCrossRefID: this parameter represents the handle provided in the Monitor Start response.
- connection: this parameter contains connection identifier of the connection in the voice browser platform that was moved to another endpoint.
- divertingDevice: this parameter provides the endpoint identifier of the diverting device on the voice browser platform.
- newDestination: this parameter represents the deflected-to device (i.e. the new device in the call).
- localConnectionState: indicates the connection state of "null" at the local connection on the voice browser platform.
- cause: indicates the cause of the event. In this case, it indicates redirected.

```
<?xml version="1.0" encoding="UTF-8"?>
<DivertedEvent xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <monitorCrossRefID>99</monitorCrossRefID>
 <connection>
      <callID>1</callID>
      <deviceID>22343</deviceID>
 </connection>
 <divertingDevice>
      <deviceIdentifier>22343</deviceIdentifier>
 </divertingDevice>
 <newDestination>
      <deviceIdentifier>333333</deviceIdentifier>
 </newDestination>
 <localConnectionInfo>null</localConnectionInfo>
 <cause>redirected</cause>
</DivertedEvent>
```

Note that the voice browser application will not see a Connection Cleared event for the diverting device after a Diverted event - the Diverted event indicates the diverting device connection has already cleared from the call.

6.14 Single Step Conference

A voice browser application can use the ECMA-323 Single Step Conference service to add another device into an existing call at the voice browser platform. As a result of the service, no devices are dropped from the call.

6.14.1 Single Step Conference - Service Request example

The mandatory Service Request parameters are:

- activeCall connection: the connection identifier of the conferencing connection (connection at the voice browser platform);

- deviceToJoin: the device identifier associated with the device to be added to the existing call;
- participationType: specifies the type of participation (listen, active, etc.) for the added device.

```
<?xml version="1.0" encoding="UTF-8"?>
<SingleStepConferenceCall xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <activeCall>
  <callID>1</callID>
 <deviceID>22343</deviceID>
 </activeCall>
 <deviceToJoin>55555</deviceToJoin>
 <participationType>active</participationType>
</SingleStepConferenceCall>
```

6.14.2 Single Step Conference - Service Response example

The telephony platform provides a positive response to the Single Step Conference service request.

The mandatory Service Request parameters are:

- conferencedCall connection: the connection identifier at the deviceToJoin's connection. Note that if the deviceToJoin is a device that is outside of the telephony platform, this connection will represent the connection of the devices associated network interface).

```
<?xml version="1.0" encoding="UTF-8"?>
<SingleStepConferenceCallResponse xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <conferencedCall>
      <callID>1</callID>
      <deviceID>55555</deviceID>
 </conferencedCall>
</SingleStepConferenceCallResponse>
```

6.15 Notification of an Party Added to a call

As the result of the Single Step Conference service, an ECMA-323 Conferenced event is provided by the telephony platform to indicate that a new device has been added to an existing call.

After the Conferenced event, additional events may be generated to reflect the progress of the call at the new device (e.g. Established event when the added device answers).

6.15.1 Conferenced Event example

The event contains the following parameters:

monitorCrossRefID: this parameter represents the handle provided in the Monitor Start response.

primaryOldCall connection: this parameter contains connection identifier of the connection of the transferring endpoint in the voice browser platform.

conferencingDevice: this parameter provides the endpoint identifier of the conferencing device on the voice browser platform.

addedparty: this parameter represents the added device (i.e. the new device in the call).

conferenceConnections: for single step conference, this includes a list of connectionIDs of the devices in the resulting call. This example shows three endpoints (22343, 33333, 55555) in the resulting call (for each endpoint its connectionID and its associated deviceID is shown in the list).

localConnectionState: this indicates the connection state at the connection on the voice browser platform. In the example, the connection state of Connected is provided.

cause: indicates the cause of the event. In this case, it indicates that the conference is a single step conference.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConferencedEvent xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
 <monitorCrossRefID>99</monitorCrossRefID>
```

```
<primaryOldCall>
    <callID>1</callID>
    <deviceID>22343</deviceID>
</primaryOldCall>
<conferencingDevice>
    <deviceIdentifier>22343</deviceIdentifier>
</conferencingDevice>
<addedParty>
    <deviceIdentifier>55555</deviceIdentifier>
</addedParty>
<conferenceConnections>
    <connectionListItem>
<newConnection>
<callID>1</callID>
<deviceID>22343</deviceID>
</newConnection>
<endpoint>
<deviceID>22343</deviceID>
</endpoint>
    </connectionListItem>
    <connectionListItem>
<newConnection>
<callID>1</callID>
<deviceID>33333</deviceID>
</newConnection>
<endpoint>
<deviceID>33333</deviceID>
</endpoint>
    </connectionListItem>
    <connectionListItem>
<newConnection>
<callID>1</callID>
<deviceID>55555</deviceID>
</newConnection>
<endpoint>
<deviceID>55555</deviceID>
</endpoint>
    </connectionListItem>
 </conferenceConnections>
<localConnectionInfo>connected</localConnectionInfo>
 <cause>singleStepConference</cause>
</ConferencedEvent>
```

Note that the voice browser application will not see a Connection Cleared event for the Transferring device after a Transferred event: the Transferred event indicates the transferring device connection has already cleared from the call.

6.16 Failure Response example

The CSTAErrorCode message is used to indicate a failure response to an ECMA-323 service request.

This example shows an error response to the Monitor Start request. The CSTAErrorCode message

The mandatory CSTAErrorCode parameters are:

- There is a choice of 7 error categories. In this example the error category of "operation" is indicated along with the specific error value of "invalidMonitorObject". This indicates that the device identifier (22343) that was provided in the Monitor Start request is invalid.

```
<?xml version="1.0" encoding="UTF-8"?>
<CSTAErrorCode xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
    <operation>invalidMonitorObject</operation>
</CSTAErrorCode>
```

Note that the CSTAErrorCode message is used to indicate a negative response for any ECMA-323 service request. See ECMA-323 clause 9.19 for a list of all possible error codes.

7 SALT/CSTA XML Programming Example

The following example is taken from the SALT 1.0 Specification. The example demonstrates the use of ECMA 323 in SALT.

The main purpose of the example is simply to ask the caller to say a phone number and transfer the call.

The SALT application can be logically composed of the following sections.

Data for the application:

```
<input name="transferTarget" />
<input name="callerID" />
<input name="callID" />
<input name="deviceID" />
<input name="monitorObject" type="hidden" value="2234" />
<input name ="monitorCrossRefID" />
```

Speech objects in English (only section affected by natural language):

```
      <listen id="recNumber" onreco  ="procRecNumber()"
onnoreco="procNoReco()
     onsilence="procNoReco()">
    <grammar src="..."/>
      </listen>

      <listen id="recYesNo" onreco="procYesNo()" onnoreco="procNoReco()"
 onsilence="procNoReco()">
    <grammar src="..."/>
      </listen>

      <prompt id="sayWelcome">Hello! Please say the phone number to
transfer to. </prompt>
      <prompt id="askAgain"> Sorry, I missed that. Please say the number
again. </prompt>
<prompt id="confirm"> Did you say <value href="transferTarget"/>?
</prompt>
<prompt id="sayBye"> Thank you. Your call is being transferred. </prompt>
      <prompt id="tryAgain">
      The number, <value href="transferTarget"/>, cannot be
    reached for transfer. Please try again later.
      </prompt>
```

Speech event handlers (dialog logic) in ECMAScript:

```
<script><!--
function procRecNumber() {
  var msg = event.srcElement.recoresult;
  transferTarget.value = msg.SelectSingleNode("*/phoneNumber").nodeValue;
     // read recognized phone number
  var confidence = msg.selectSingleNode("/@confidence").nodeValue;
  if (confidence < 0.5) {
     confirm.Start(); recYesNo.Start();
  } else {
     sayBye.Start(); ccTransfer();
  }
}

function procYesNo() {
  var answer = event.srcElement.recoresult.SelectSingleNode(
       "*/yes[@confidence>0.5]");
         // accept only yes with confidence
```

```
 if (answer == null) {
   procNoReco();
 } else {
   sayBye.Start(); ccTransfer();
 }
}

function procNoReco() {
 transferTarget.value = "";
 askAgain.Start(); recNumber.Start();
}
--></script>
```

The call control section (unaffected by locale, dialog logic):

```
<smex id="callControl" onreceive="ccHandler()">...</smex>
<script><!--
      // The cchandler handles the ECMA-323 events.
      //
      // Once the connection is answered, a welcome prompt
      // is played and the transfer target telephone number is solicited.
      //
      // When the speech event handler detects and confirms the correct
      // speech input, an ECMA-323 SingleStepTransfer service is used to
      // transfer the caller to the new transfer target.
      //
function ccHandler() {
 var msg = event.srcElement.received;
 if (msg.nodeName == "DeliveredEvent") { // incoming call notification
      //
      // If the connection is alerting (DeliveredEvent, ECMA-323, 15.2.5)
the
      // connection information from the Delivered event is saved
      // called.value and deviceID.value) and the call is answered by
using the
      // ECMA-323 AnswerCall service with the saved connection
information.
      // If the application needed the ANI and DNIS, it could also obtain
      // this information from this event.
      //
            callID.value = msg.selectSingleNode(
            "./connection/callID").nodeValue;
   deviceID.value = msg.selectSingleNode(
                  "./connection/deviceID").nodeValue;
   ccAnswer();
 } else if (msg.nodeName == "EstablishedEvent") { // call answered
      //
      // Once the connection is answered (EstablishedEvent, ECMA-323,
15.2.8)
      // a welcome prompt is played and the transfer target telephone
number
      // is solicited.
      //
   callerID.value = msg.selectSingleNode(
     "./callingDevice/DeviceIdentifier").nodeValue;
   sayWelcome.Start(); recNumber.Start();
 } else if (msg.nodeName == "TransferredEvent") { // call transferred
      //
      // The TransferredEvent (ECMA-323, 15.2.18) is received when
      // the transfer has been completed. ccCleanup is called to clean up
      // the application data.
      //
```

```
    ccCleanUp();
  } else if (msg.nodeName == "ConnectionClearedEvent") { // user hang up
       //
       // A user hang up is indicated by a ConnectionClearedEvent
(ECMA-323,
       // 15.2.4) which flushes the prompt queue and cleans the application
       // data. This could happen at any time during the call.
       //
    promptQueue.Flush();
    ccCleanUp();
  } else if (msg.nodeName == "CSTAErrorCode") { // service failure event
       //
       // The ccError function handles any failure responses from any of
the
       // ECMA-323 services that may have failed.
       //
             ccError();
       } // feel free to handle other events here
}

function ccTransfer() { // transferring a call
  //
  // The SingleStepTransferCall service (ECMA-323, 15.1.24) is used to
  // invoke the transfer. There are two elements provided. The first
  // element is the connection information that was obtained from
  // the DeliveredEvent. The second element is the transfer target that
  // was solicited from the caller.
       //
  callControl.sent = "<SingleStepTransferCall" +
             " xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>" +
             "<activeCall><callID>" +
    callID.value + "</callID><deviceID>" +
    deviceID.value + "</deviceID></activeCall><transferredTo>" +
    transferTarget.value+"</transferredTo></SingleStepTransferCall>";
}

function ccStartListening() { // listening for call events
  //
  // The MonitorStart service (ECMA-323, 13.1.2) is used to place a
  // monitor on a device so that events can be generated when activity
  // happens at that device. The single element provided indicates the
  // identifier of the device that is to be monitored. In this example
  // it was part of the application data.
  //
  callControl.sent = "<MonitorStart" +
             " xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>" +
    "<monitorObject><deviceObject>" +
             monitorObject.value +
      "</deviceObject></monitorObject></MonitorStart>";
}

function ccAnswer() { // answering a call
  //
  // The AnswerCall service (ECMA-323, 15.1.3) is used to answer the
  // alerting connection. The single element provided is the connection
  // information that was obtained in the Delivered event.
  //
  callControl.sent = "<AnswerCall" +
             " xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>" +
    "<callToBeAnswered><callID>" +
             callID.value + "</callID><deviceID>"
             deviceID.value +
             "</deviceID></callToBeAnswered></AnswerCall>";
```

```
}

function ccCleanUp() {
 callerID.value = ""; transferTarget.value = ""; callID.value = "";
 recNumber.Stop(); recYesNo.Stop(); ...
}

function ccHangup() { // clearing a connection
 //
 // The ClearConnection service (ECMA-323, 15.1.8) is used to clear
 // a connection. In this example, this is used when the transfer is
 // unable to be completed.
 //
 callControl.sent = "<ClearConnection " +
            "xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>" +
            "<connectionToBeCleared> <callID>" +
            callID.value + "</callID><deviceID>" +
            deviceID.value + "</deviceID></connectionToBeCleared>" +
            "</ClearConnection>";
}

function ccError() {
 //
      // The ccError function is called to handle any failure responses to
 // ECMA-323 service requests. If there was an error starting a
 // monitor, the application logs an error. If there was an error
 // response to the SingleStepTransfer service, a message is played
 // for the caller and the connection is cleared.
 //
 var request = callControl.sent.substr(1, 7);
      // read the first 7 characters of service requested
 if (request == "Monitor") { // error starting a monitor
    logMessage("ccError", callControl.sent);
 } else if (request == "SingleS") { // error in transfer
    tryAgain.Start();
    ccHangup();
 } // feel free to handle other errors here
}
--> </script>
```

Putting it all together:

```
<html>
...
<body>
// data section here
<div xmlns="http://www.saltforum.org/2002/SALT" style="visibility:hidden">
      // put the speech objects here
</div>
// speech event handlers here
// call control section here
// finally, when the page is loaded...
<script>
 ccStartListening();
</script>
</body>
</html>
```

8 CCXML/CSTA XML Programming Example

The following example demonstrates the use of ECMA-323 with CCXML. The example shows how the ECMA-323 services and events used in the previous example are created and exposed via CCXML.

```
<?xml version="1.0" encoding="UTF-8"?>

<ccxml version="1.0"
xmlns="http://www.w3.org/2002/09/ccxml"
xmlns:ccxml="http://www.w3.org/2002/09/ccxml"
xmlns:csta="http://www.ecma.ch/standards/ecma-323/csta/ed2">

 <var name="state0"/>

 <!-- CSTA Vars -->
 <var name="callID"/>
 <var name="deviceID"/>

 <eventhandler statevariable="state0">

 <transition event="ccxml.loaded">

 <!-- inline ECMA-323 -->
 <csta:MonitorStart>
      <monitorObject>
        <deviceObject>
          9999
        </deviceObject>
      </monitorObject>
   </csta:MonitorStart>

   <!-- ECMA-323 as a external event source -->
 <script>
      var monitorRequest = "<MonitorStart"
        + " xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>"
        + "<monitorObject><deviceObject>" + 9999
        + "</deviceObject></monitorObject></MonitorStart>";
```

```
</script>
<send dest="http://www.csta-webservice.com/ecma323" name="sendRequest"
namelist="monitorRequest" />

</transition>

<transition event="csta.DeliveredEvent" name="evt">
<!-- Simple CCXML style shortcut -->
   <accept/>

   <!-- save off event props -->

   <!-- simple object mapping -->
   <assign name="callID" expr="evt.connection.callID"/>
   <assign name="deviceID" expr="evt.connection.deviceID"/>

<!-- DOM API -->
<script>
     callID.value = evt.selectSingleNode("./connection/callID").value;
     deviceID.value =
     evt.selectSingleNode("./connection/deviceID").value;
</script>

<!-- Big inline ECMA-323 style request -->
   <AnswerCall xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
     <callToBeAnswered>
     <callID> <ccxml:value expr="evt.connection.callID"/> </callID>
     <deviceID> <ccxml:value expr="evt.connection.deviceID"/>
     </deviceID>
     </callToBeAnswered>
   </AnswerCall>

   <!-- ECMA-323 as a external event source -->
<script>
     var answerRequest = "<AnswerCall"
```

```
        + " xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>"
        + "<callToBeAnswered><callID>" + evt.connection.callID
        + "</callID><deviceID>" + evt.connection.deviceID +
        "</deviceID></callToBeAnswered></AnswerCall>";
 </script>
 <send dest="http://www.csta-webservice.com/ecma323" name="sendRequest"
namelist="answerRequest" />


 </transition>


 <transition event="csta.EstablishedEvent" name="evt">
 <dialogstart src="'hello.vxml'"/>
 </transition>


 <transition event="dialog.exit">
 <disconnect/>


 <!-- Big inline ECMA-323 style request -->
    <ClearConnection xmlns="http://www.ecma.ch/standards/ecma-
323/csta/ed2">
      <connectionToBeCleared>
        <callID><ccxml:value expr="callID"/></callID>
        <deviceID><ccxml:value expr="deviceID"/></deviceID>
      </connectionToBeCleared>
    </ClearConnection>


    <!-- ECMA-323 as a external event source -->
    <script>
      var clearRequest = "<ClearConnection"
        + " xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>"
        + "<connectionToBeCleared><callID>" + evt.connection.callID
        + "</callID><deviceID>" + evt.connection.deviceID +
        "</deviceID></connectionToBeCleared></ClearConnection>";
 </script>
 <send dest="http://www.csta-webservice.com/ecma323" name="sendRequest"
    namelist="clearRequest" />
```

```
 </transition>

 <transition event="csta.ConnectionClearedEvent" name="evt">
 <exit/>
 </transition>

 <transition event="error.*" name="evt">
 <log expr="'an error has occurred (' + evt.error + ')'"/>
 <exit/>
 </transition>
 </eventhandler>
</ccxml>
```

9 CSTA Call Control Features

Some ECMA-323 conformant platforms may provide more advanced call control features. This clause provides an overview of the complete set of CSTA Call Control Features. Please refer to ECMA-269 for a detailed specification of the features.

9.1 Services

- Accept Call: Causes an offered call to transition to the Ringing or Entering Distribution mode of the alerting state.
- Alternate Call: Places an existing call on hold and then retrieves a previously held or alerting call at the same device.
- Answer Call: Answers a call that is ringing, queued, or being offered to a device.
- Call Back Call-Related: Allows a computing function to request that an originally called device return a call to the original calling device.
- Call Back Message: Call Related: Allows a computing function to instruct the switching function to leave a pre-defined message requesting that the called device call the calling device.
- Camp On Call: Queues a call at a busy device until the device becomes available.
- Clear Call Releases all of the devices associated with the specified call.
- Clear Connection: Releases a specific device from a call.
- Conference Call: Provides a conference of an existing held call and another active call at a conferencing device. The two calls are merged into a single call at the conferencing device.
- Consultation Call: Places an existing active call at a device on hold and initiates a new call from the same device.
- Deflect Call: Deflects a call to another device.
- Dial Digits: Dials a digit sequence for a call that has already been initiated.
- Directed Pickup Call: Picks a specified call. (Moves and connects a specified alerting or queued call).

- Group Pickup Call: Picks a call from a specified pick group. (Moves and connects any alerting call in a pick group to another device.)
- Hold Call: Places a specific connection on hold.
- Intrude Call: Allows a computing function to add the calling device to a call at a busy called device.
- Join Call: Allows a computing function to request, on behalf of a device, that the device be joined into an existing call.
- Make Call: Establishes a call between two devices.
- Make Predictive Call: Establishes a call between two devices. The calling device is presented with the call only after the called device is alerted or has answered the call.
- Park Call: Parks a call at a specified device. (Moves and queues a connected call to another device).
- Reconnect Call: Clears an existing connection and then connects a previously held connection at the same device.
- Retrieve Call: Connects to a call that had previously been placed on hold.
- Single Step Conference Call: Adds a device to an existing call.
- Single Step Transfer Call: Replaces a device in an existing call with another device.
- Transfer Call: Transfers a held call to the consulted party.

9.2 Events

- Bridged: Indicates that an appearance at a shared bridged device configuration has been placed into an inactive mode (i.e. queued state).
- Call Cleared: Indicates that all devices have been removed from an existing call.
- Conferenced: Indicates that the conferencing device has conferenced itself or another device with an existing call.
- Connection Cleared: Indicates that a device in a call has disconnected or dropped out from a call.
- Delivered: Indicates that a call is being presented to a device in either the Ringing or Entering Distribution modes of the alerting state.
- Digits Dialled: Indicates that a call or feature is being attempted from a device and that a portion of the dialling sequence has been completed.
- Diverted: Indicates that a call has been diverted from a device.
- Established: Indicates that a device has answered or has been connected to a call.
- Failed: Indicates that a call cannot be completed and/or a connection has entered the Fail state.
- Held: Indicates that an existing call has been put on hold.
- Network Capabilities Changed: Indicates that a situation occurred during a call's progress in a public or private network that modifies its signalling capability (i.e. inter-networking).
- Network Reached: Indicates that a call has been connected to an external network using a Network Interface Device (e.g., trunk, CO Line).
- Offered: Indicates that a call is in a pre-delivery state at a device (prior to ringing indication or delivering ringback, for example).
- Originated: Indicates that a call is being attempted from a device.

- Queued: Indicates that a call has been queued.
- Retrieved: Indicates that a previously held call has been retrieved.
- Service Initiated: Indicates that a device has gone off-hook for service or is being prompted to go off-hook.
- Transferred: Indicates that an existing call has been transferred to another device and that the device transferring the call has been dropped from the call.

History

Document history		
V1.1.1	February 2003	Publication