Workshop on speaker biometrics and VoiceXML 3.0
March 5-6, 2009, Menlo Park, CA, US

Proposal of an SIV architecture and requirements

Authors
Martin Eckert (martin.eckert@telekom.de), Ingmar Kliche (ingmar.kliche@telekom.de), Deutsche Telekom Laboratories.

Abstract
Deutsche Telekom, the largest European telecommunication operator, is using speaker verification technology in various services (like password reset) since a couple of years. These services have been implemented based on proprietary speaker identification and verification (SIV) technology in the past. But integration of SIV technology into more services, especially in large deployments, requires standardization: Application development need to be independent (to a certain extend) from the underlying (vendor specific) SIV technology and SIV engines need to be exchangeable. We believe that VoiceXML is the ideal basis to be the API for developing SIV applications. Hence VoiceXML should be extended to support SIV. This paper proposes an SIV architecture and derives various functional requirements.

1. Introduction

Over the past few years various use cases and requirements for SIV standardization have been discussed, mainly driven by the VoiceXML Forum, Speaker Biometrics Committee. It appears to us that the large number of use cases and possible implementation variants (e.g. SIV combined with ASR vs. SIV and ASR as independent components) are a burden for a successful standardization. Therefore we propose to reduce the number of possibilities to a reasonable amount and to start standardization of SIV within the VoiceXML 3.0 framework. This position paper proposes architecture and a set of requirements for a first version of SIV.

2. Proposed Architecture

A typical VoiceXML system contains main components like telephone network (PSTN or VoIP packet network) interface, VoiceXML gateway and application server. The VoiceXML gateway mainly consists of a VoiceXML interpreter (VoiceXML Browser) and media resources like ASR, TTS, Play/Record, DTMF recognizer etc. For SIV systems an SIV engine must be integrated. The Interface between VoiceXML Browser and media resources can be a native API or use standards like MRCP (Media Resource Control Protocol). The application server is typically a web server that hosts the VoiceXML application. Speaker verification or identification (SIV) is performed by comparing so called voiceprints with utterances (or feature sets) taken from the current user (caller). Since voiceprints are typically stored in databases a voiceprint database has to be added to the VoiceXML architecture to run SIV.

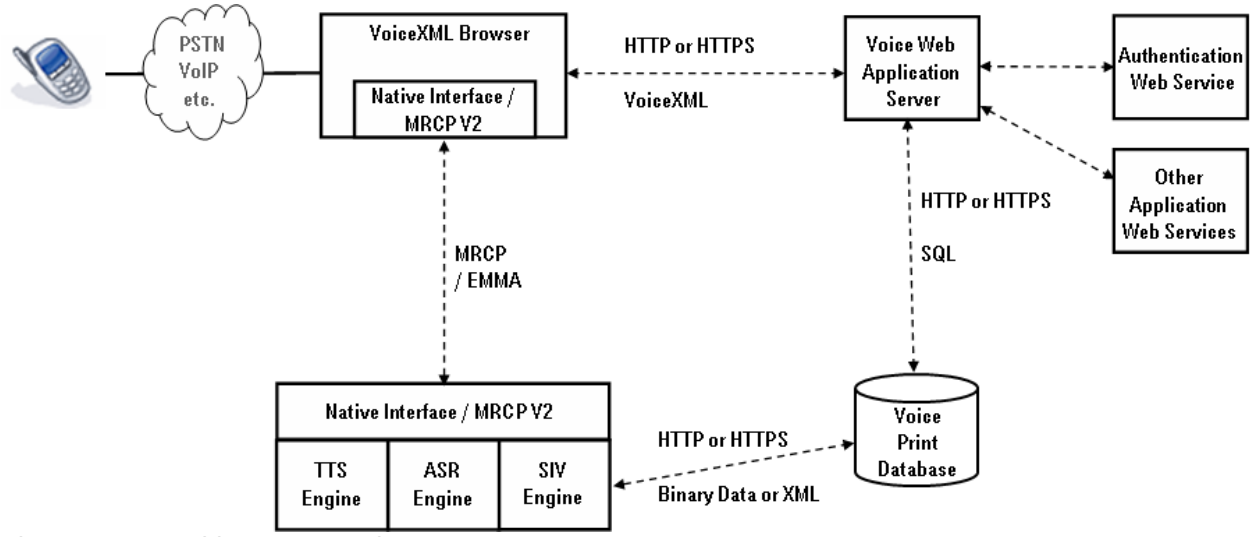

Figure 1: SIV architecture overview

VoiceBrowser/SIV engine interface
MRCP, the Media Resource Control Protocol (developed by the IETF), is a communication protocol which allows speech servers to provide various speech services (e.g. speaker verification and speech recognition) to its clients. MRCP is the de facto standard for integration of speech resources in VoiceXML systems and is currently specified in version 2, which provides messages for SIV functionalities.

Proposal: We propose to support the use of MRCP for integration of SIV engines with VoiceXML browsers. From an operators point of view a separation of platform and resources is preferred and is necessary especially for large deployments. We would like to discuss, whether the set of functionalities of SIV within VoiceXML 3.0 could be limited to those supported by MRCP v2.

Some SIV engine vendors already provide an MRCP V2 server that allows controlling the SIV engine via MRCP.

Restrictions when using MRCP V2
Currently there is no way to pass voiceprints within MRCP messages, but only URLs referring to voiceprints are allowed. The voice application has to provide a reference to the voiceprint and can not provide the voiceprint itself. It is the responsibility of the SIV engine to load the voiceprint from the given URL.

This given restriction leads to the proposed architecture above, which contains a connection from the SIV engine to the voiceprint database.

Some SIV vendors use proprietary SIV MRCP extensions (messages), since MRCP v2 does not support all functionalities which have been implemented by some SIV engines vendors.

Proposal: For maximum interoperability proprietary extensions to MRCP should not be used. Instead of using proprietary messages we propose to extend MRCP if necessary (e.g. after analyzing the missing pieces of MRCP v2).

Voiceprint management (storage and transport)
Voiceprints are usually stored in a voiceprint database. SIV engines need to load voiceprints from the database for verification/identification and save/store them into the database for enrollment and adaptation.

The following method for voiceprint transport between SIV engine and voiceprint database is proposed:

Proposal: The voiceprint URL is provided by the application via VoiceXML. The VoiceXML browser sends the voiceprint URL within MRCP messages to the SIV engine and the SIV engine uses the URL

to fetch the voiceprint from the voiceprint database. Hence the SIV engine needs a connection to the voiceprint database.

The transport protocol between SIV engine and voiceprint database should support

1) security and
2) different deployment scenarios (including a scenario where the SIV engine is running on a mobile device and the voiceprint database is a central function residing in a network).

Proposal: Therefore we propose to consider the use of the HTTP/HTTPS protocol. It supports both, security and distribution of components within the network.

In this case the voiceprint database will be essentially a web service.

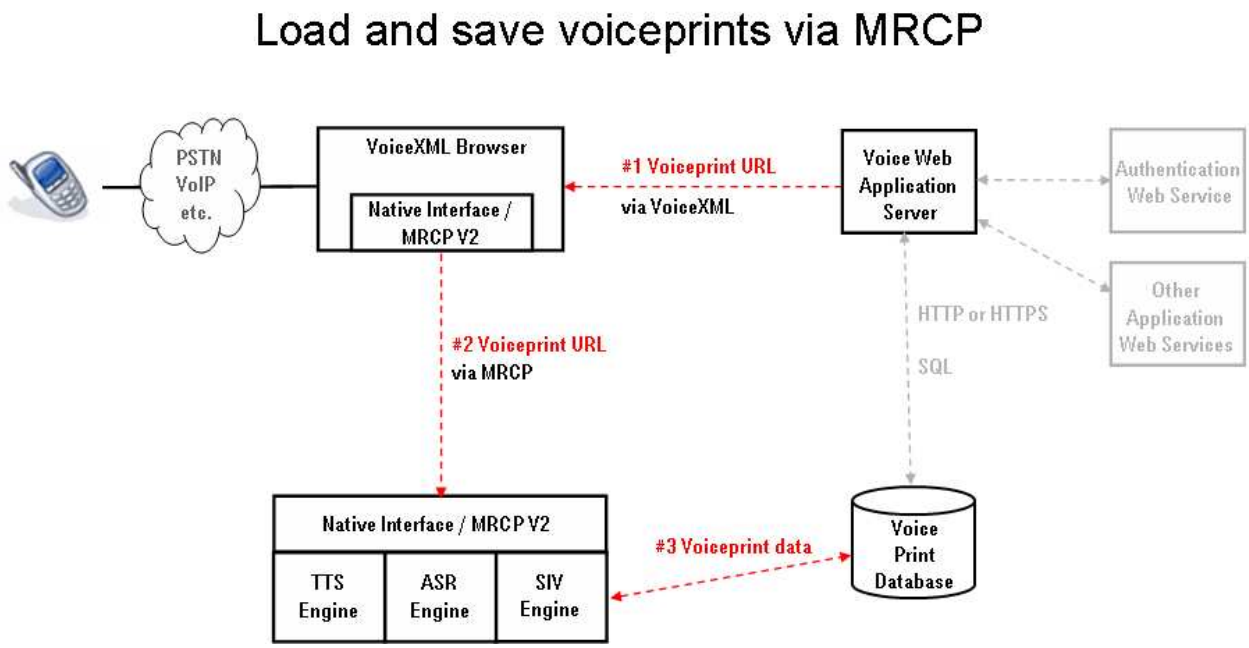

Figure 2: Querying of voiceprint database from SIV engine

Voiceprint management also requires functions to copy, delete and query voiceprints. There are two options for copy, delete and query functionality:

1) Executed by SIV engine (commands send from VoiceXML browser via MRCP messages)
2) Executed by the voice application either using common database functions (SQL etc.) or HTTP requests (which lead to execution of native database actions within the voiceprint database)

MRCP V2 provides copy, delete and query functions (messages). But the second option is preferred for performance reasons. Otherwise every query or copy initiated by the voice application would have to be rendered into VoiceXML, executed by the VoiceXML browser and sent to the SIV engine using MRCP/SIV commands.

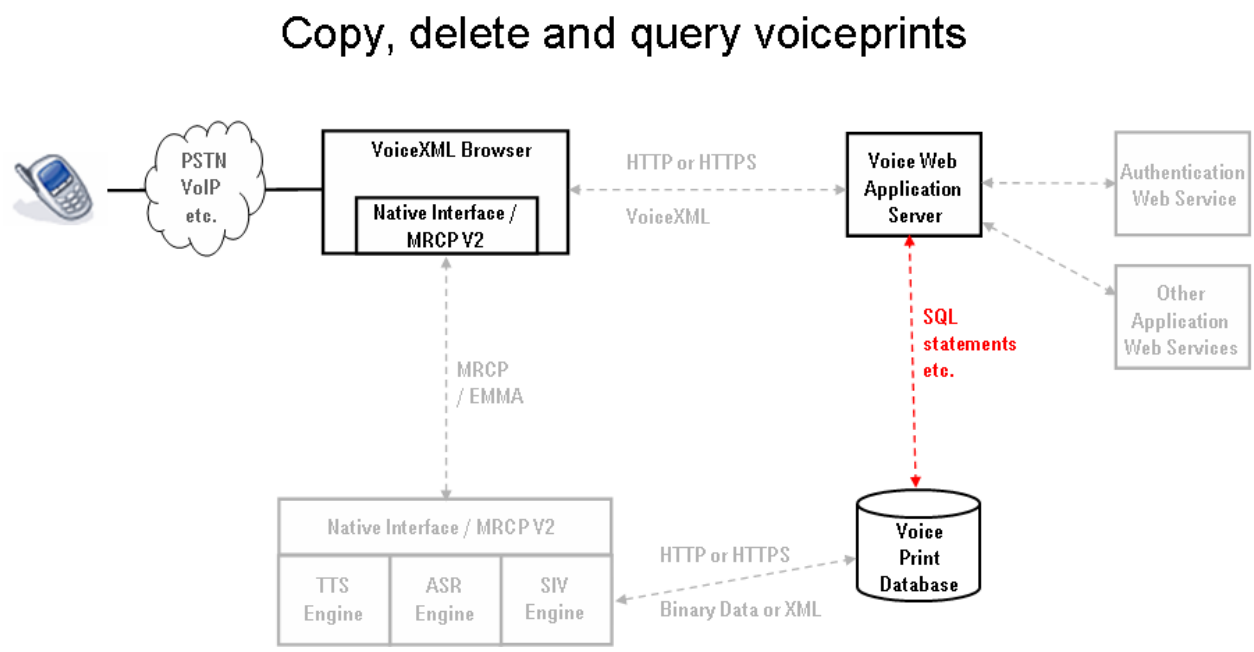

Figure 3: Querying of voiceprint database from voice application

EMMA
EMMA, the Extensible Multimodal Annotation markup language, has been designed to annotate and describe user input. We expect a wide adoption of EMMA within ASR implementations and VoiceXML 3.0 in the near future.

Proposal: We have a strong preference to use EMMA for representation of results of SIV functions, like decision, score/confidence etc.

3. SIV requirements and core functionalities

This chapter collects functionalities and requirements based on experiences from working with various SIV products from different vendors. We also try to describe some terms used within SIV technology to create a commons understanding.

Combination of SIV with other resources (especially ASR)
In general SIV can be performed in three ways:

1) SIV only (i.e. without ASR, standalone SIV)
2) SIV in parallel to ASR (ASR and SIV are separate resources)
3) SIV integrated with ASR as one (combined) resource

Proposal: The number of options should be reduced to

1) standalone SIV and
2) SIV / ASR as separate resources

for a first version of SIV support in VoiceXML 3.0. Future versions of VoiceXML or of the SIV module within VoiceXML may also add support for option 3).

Decision control
During an SIV session (e.g. verification) either the SIV engine or the applications have to take decisions (about acceptance/rejection). Both approaches are in use today by different products of different vendors.

Proposal: Both approaches should be supported by the SIV functionality within VoiceXML 3.0.

Types of SIV
There are three different SIV approaches available:

- Text independent: SIV technology that can operate on any freeform or structured spoken input
- Text dependent: SIV technology (usually verification) that requires the voice input of one or more specific passwords or pass phrases (having been enrolled).
- Text prompted: SIV technology (usually verification) that randomly selects words and/or phrases and prompts the speaker to repeat them. The term is also called challenge-response[1].

Proposal: We propose that all of these types should be supported.

SIV-phases and sessions
SIV consists of two phases:

1) Enrollment (also known as training)
2) Verification/Identification.

Phase 2 (verification/identification) is only available after successful enrollment (Phase 1). Enrollment and Verification/Identification itself are session based. They usually contain various turns.

The following figure shows the relation of a SIV session and a dialog session. An SIV session is usually a part of a dialog session.

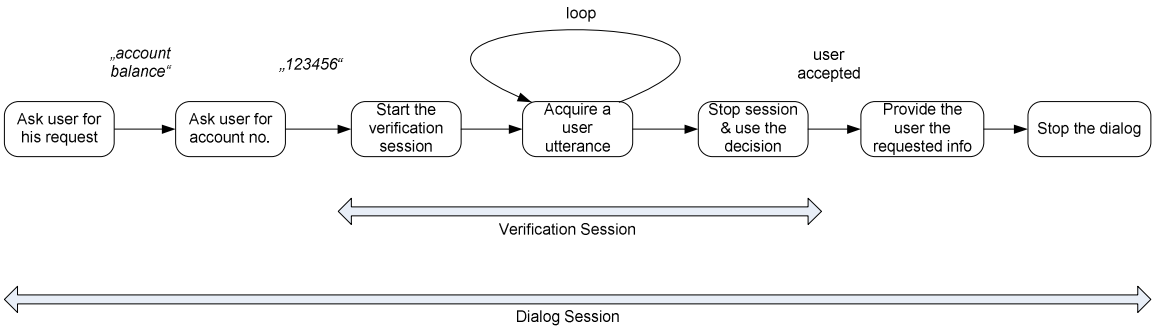

Figure 4: Relationship of verification and dialog session

Enrollment or Verification/Identification sessions are typically performed over one or more turns. The following figure shows the flow of an enrollment session:

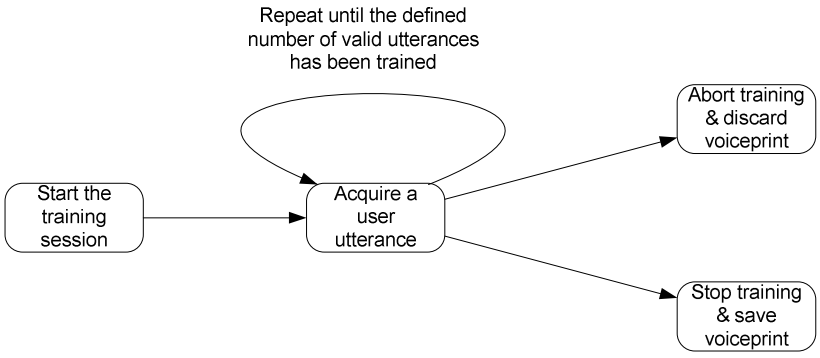

Figure 5: SIV enrollment session

The following figure shows the flow of a verification session:

[1] Source: VoiceXML Forum SIV Glossary

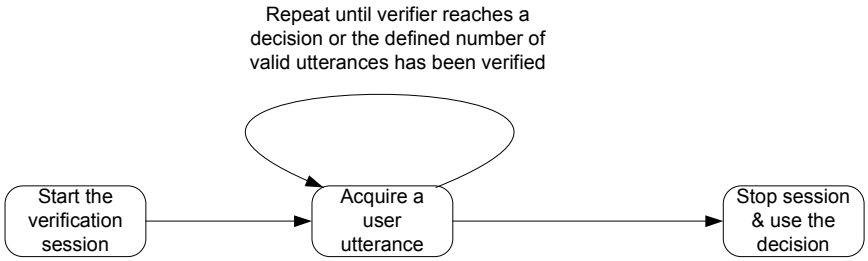

Figure 6: Verification session

SIV sessions (either enrollment or verification/identification) are started and (usually) stopped by the application. But there are also cases where SIV sessions may be stopped by the SIV engine (e.g. decision reached, errors etc.) (see decision control chapter).

Buffering of user utterances for later use
In various application scenarios it is useful or even necessary to apply SIV functions to a user utterance in a later dialog stage. For example the user might enter his identity claim (e.g. a customer number) using ASR. The recognition result will be used to access the voice print. To allow applying a verification check even to the user utterance (which contained the user id) it is necessary to buffer the utterance.

This functionality might be implemented by recording the utterance in parallel to the recognition task and applying the SIV function to the recorded utterance later. Then VoiceXML 3.0 would need to support “SIV from file” functionality.

Adaptation of voiceprints
Adaptation is a process for increasing speaker recognition accuracy by updating the voiceprint during verification (sometimes also identification).
If the user is accepted and the verification score exceeds a certain adaptation threshold the current user utterance is used after verification to “retrain” or “refresh” the voiceprint.

The application must be able to control the adaptation process. Usually this is done by simply turn adaptation on or off within verification/identification phase.
.
Rollback/Undone of last turn
When performing text-dependant or text-prompted verification sometimes ASR (parallel to SIV) is used to check whether the user really spoke the prompted (required) utterance. In the case the ASR recognizes a wrong user input a roll back of the last (wrong) turn has to be performed to make sure this turn is not used for verification.

That’s why a mechanism to “undo” or roll back the processing performed on the last turn (and only the last turn) in the training and verification/identification session should be supported by a VoiceXML 3.0 SIV application.

Basic/core functionalities for application development
The following list of core functions (which have been discussed in the previous chapters) should be provided to the application developer and hence need to have a representation in VoiceXML 3.0:
- Enrollment, Verification, Identification
- Query/Copy/Delete of voiceprints
- Buffering of user utterances for later use
- Adaptation of voiceprints
- Rollback/Undone of last turn
- Query SIV results (e.g. accept/reject information, score etc.)
- Catch SIV events (e.g. noinput or nomatch events)

Note that “Buffering of user utterances for later use” might be implemented using other core functionalities like recording and later application of SIV functions onto the recorded utterance. But this requires that “SIV from file” is somehow supported with the VoiceXML 3.0 syntax.